FUNDAMENTAL DESIGN ISSUES IN ANONYMOUS PEER-TO-PEER DISTRIBUTED HASH TABLE PROTOCOLS

A DISSERTATION SUBMITTED TO THE GRADUATE DIVISION OF THE UNIVERSITY OF HAWAIʻI AT MĀNOA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE

AUGUST 2019

By

Todd A. Baumeister

Dissertation Committee:

Yingfei Dong, Co-chairperson
Edoardo S. Biagioni, Co-chairperson
Dusko Pavlovic
Peter-Michael Seidel
Yao Zheng

To my wife, Andrea.

This is very much your fault. Thanks.

ACKNOWLEDGMENTS

This material is based upon work supported by the NSF grant CNS-1041739.

I would like to extend my sincere thanks to my advisor Professor Yingfei Dong for his guidance, patience, and all of the research opportunities that he provided me. I would also like to thank our colleagues Guanyu Tian and Zhenhai Duan from Florida State University for their contributions and collaboration on the protocol analysis. In addition, I would also like to thank Dwayne Yuen for his contributions to our experiment test bed.

I am also grateful to my dissertation committee Professor Edo Biagioni, Professor Depeng Li, Professor Dusko Pavlovic, and Professor Yao Zheng for the feedback they provided. Finally, I would like to acknowledge all of my family and friends for their support and encouragement.

ABSTRACT

Anonymous communication protocols can be used to protect diminishing user privacy online. One family of those protocols is anonymous peer-to-peer (ANP2P) distributed hash table (DHT). These protocols are considered efficient, scalable, decentralized, and practical. However, the anonymity properties of these protocols are not well understood and difficult to quantify. As a result, users may have a false sense of anonymity when using these protocols. This motivates our study of the anonymity properties of ANP2P DHT protocols.

In this study, we analyzed the Freenet and GNUnet systems. We cataloged the main design decisions and identified three vulnerabilities in these ANP2P DHT protocols. The first vulnerability was the *Traceback attack*, which enables an adversary to determine which subset of nodes routed a given message. The second vulnerability was the *Routing Table Insertion attack*, which can be used by an adversary to place themselves in a victim's routing table. We developed this attack to support the Traceback attack, and we present two potential mitigations - routing randomness and Look Ahead Hint. The third vulnerability was an attack on GNUnet's message bloom filter. Similar to the Traceback attack, the GNUnet bloom filter vulnerability can be used to determine which subset of nodes routed a given message.

We then developed an empirical methodology for modeling a generic adversary and evaluating the performance and anonymity of ANP2P DHT design decisions. We created a novel adversarial model that uses protocol behaviors and shared states to sample the entropy in an ANP2P DHT system. The adversarial model implements a routing path *Walk-back Ranking Algorithm* that can be used to identify potential sender nodes for a given message. The methodology was then applied to an extension of the peer-to-peer network simulator PeerSim. We extended PeerSim to implement various ANP2P DHT design decisions. We then used the extended PeerSim and the methodology to evaluate anonymity and performance for structured, small-world, and random topologies using look ahead values of one-hop and two-hop. These experiments also validated the accuracy of the methodology. Next, we used the output of the methodology to provide a quantitative comparison of the performance and anonymity for the design decisions. The methodology was also able to identify several network sub-graphs that degraded anonymity in small-world topologies.

Protocol designers can use our methodology to evaluate anonymity and performance of their protocols, and to identify the existence of network sub-graphs that degrade anonymity. Once these sub-graphs have been identified, protocol designers can create appropriate controls to prevent their formation. Future work includes applying our empirical methodology to the mitigations we proposed for the Routing Table Insertion attack and evaluating the methodology on a real-world protocol.

TABLE OF CONTENTS

Acknowledgments . . . . . **iv**

Abstract . . . . . **v**

List of Tables . . . . . **ix**

List of Figures . . . . . **x**

1 Introduction **1**

1.1 User Privacy Concerns 1

1.2 Anonymous Peer-to-peer Protocols 1

1.3 Contributions 3

1.4 Organization 4

2 Background and Related Work **6**

2.1 Anonymity Definitions 6

2.2 Anonymous Peer-to-peer Protocols 7

2.2.1 Anonymous Peer-to-peer Protocol Categories 8

2.2.2 History of Anonymous Peer-to-peer Protocols 9

2.3 Anonymity Metrics 13

2.3.1 Process-calculi 14

2.3.2 Probability-Based Methods 14

2.3.3 Information Theory 14

2.4 Anonymity Analysis of ANP2P Data Sharing Protocols 16

2.5 Open Research Issues 17

2.5.1 Secure Peer Selection 17

2.5.2 Secure Message Forwarding . . . 18
2.5.3 Sybil Attack . . . 18
2.5.4 Performance . . . 19
2.5.5 Secure Assignment of Node Identity . . . 20
2.5.6 Provable Anonymity . . . 21
3 Analysis Assumptions . . . 22
3.1 Adversary . . . 22
3.1.1 View . . . 22
3.1.2 Behavior . . . 22
3.1.3 Position . . . 23
3.1.4 Colluding . . . 23
3.1.5 Conclusion . . . 23
3.2 Anonymity . . . 24
3.2.1 Sender Anonymity . . . 24
3.2.2 Anonymity Sources . . . 24
4 Anonymous P2P DHT Protocol Specification Analysis . . . 26
4.1 Methodology . . . 26
4.2 Freenet Overview . . . 27
4.3 Design vs. Implementation . . . 29
4.4 Traceback Attack . . . 29
4.5 Routing Table Insertion Attack . . . 32
4.5.1 Routing Prediction Model . . . 34
4.5.2 Effectiveness of the RTI Attack . . . 36

4.5.3 Attack Pair Analysis . . . 37
4.5.4 Examining the coverage of the RTI Attack . . . 38
4.5.5 Evaluation of Route Prediction Model . . . 38
4.5.6 Mitigations . . . 40
4.6 GNUnet Bloom Filter . . . 52
4.7 Summary . . . 53
5 Design Choices . . . 55
6 Empirical Approach to Design Analysis . . . 60
6.1 Experiment Setup . . . 60
6.1.1 Metrics . . . 62
6.2 Simulation . . . 66
6.3 Results . . . 67
6.3.1 Experiment Assumptions . . . 67
6.3.2 Performance . . . 68
6.3.3 Anonymity . . . 71
6.4 Summary . . . 78
7 Conclusions . . . 80
7.1 Summary of Contributions . . . 80
7.2 Future Work . . . 81
A Freenet Protocol . . . 82
A.1 CHK Request Data . . . 82
Bibliography . . . 84

LIST OF TABLES

4.1 Reconstructed routing table of node H. 34

4.2 Predicted routes from origin node H with a length of three. 36

4.3 Maximum probability of uniquely identifying a node from Look Ahead Hint in our experiments in Figure 4.17. 51

5.1 Design Choices used in empirical analysis 56

6.1 Empirical Study of Design Choices 61

6.2 Mean Routing Path Length and Percentage of Path Lengths 50 or Greater 70

LIST OF FIGURES

2.1 Anonymous peer-to-peer overlay network. 8

2.2 Anonymous access overlay network. 9

2.3 Anonymous data sharing overlay network. 9

2.4 Time line of creation dates for a sample of ANP2P protocols. 10

4.1 Freenet Routing. Clarke I, others. A distributed decentralized information storage and retrieval system. Undergraduate Thesis. 1999. 30

4.2 Traceback attack. 31

4.3 Illustration of Routing Table Insertion (RTI) attack. 33

4.4 Example topology. Node location in parentheses. 35

4.5 Effectiveness of RTI attacks on a 125-node network. 36

4.6 Percentage distribution of the shortest distance between attack nodes when the maximum RTI attack distance is seven. The best attack nodes and worst attack nodes are shown. 37

4.7 RTI Attack Coverage. (With 10% attack nodes, more than 50% nodes become victims on random topologies.) 39

4.8 Average number of attack resources needed to target 100% of the nodes in a network using the attack pairs with the minimum coverage. 39

4.9 Accuracy of our route prediction model compared to actual routes by HTL. 40

4.10 Routing performance in random topologies with one-hop look ahead. 43

4.11 Routing performance in random topologies with two-hop look ahead. 43

4.12 Routing performance in small-world topologies with one-hop look ahead. 44

4.13 Routing performance in small-world topologies with two-hop look ahead. 44

4.14 Example of look ahead hint (LAH) of size one. 46

4.15 Look ahead routing performance on small-world and random topologies. 48

4.16 Estimated adversary resources needed to construct complete real-time topology. . . . 49

4.17 LAH routing performance on small-world topologies with two-hop look ahead. 50

4.18 False positive probability of a GNUnet message Bloom filter. 53

6.1 Example Structured Topology with 40 nodes 62

6.2 Example structured topology with 20 nodes. 64

6.3 Routing Tree of Top Candidates 66

6.4 CDF of Routing Path Lengths by Topology Type and Look Ahead 69

6.5 Box Plot of Routing Path Lengths by Topology Type and Look Ahead 69

6.6 Accuracy of Top-rank Set by Topology Type and Look Ahead 72

6.7 Mean Top-rank Set Entropy by Topology Type and Look Ahead 73

6.8 Example Top Ranked Routing Path Tree with Adversary Distance of 8 hops 74

6.9 Example Mean Top-rank Set Entropy for Example 100 node Structured Topology . . 75

6.10 Minimum Top-rank Set Entropy by Topology Type and Look Ahead 77

6.11 Adjusted Mean Top-rank Set Entropy by Topology Type and Look Ahead 78

A.1 State transition diagram: request CHK data. 82

A.2 Sub state transition diagram: return CHK data X. 82

A.3 Sub state transition diagram: CHK data not found. 83

CHAPTER 1
INTRODUCTION

1.1 User Privacy Concerns

Diminishing online privacy is an important issue on today's Internet. We are constantly connected with our computers and mobile devices, and this connectivity generates a wealth of information about our online behaviors. If collected, this information could compromise personal privacy or be used for digital surveillance. One solution is to create new laws and regulations to protect our online privacy. However, laws and regulations will not completely solve the problem because some organizations will strategically place their resources in countries that have weak cyber laws, making it difficult or impossible to enforce online privacy laws internationally. Another solution is to use tools that support anonymity. Anonymous communication protocols allow users to hide their identities while communicating with others. The use of these anonymous protocols can help safeguard our online privacy by hiding our actions and identities within a crowd of users. This motivates our study on the anonymity properties of anonymous peer-to-peer (ANP2P) distributed hash table (DHT) protocols.

There are a few main reasons that online privacy is diminishing. Decreasing digital storage and computation costs are allowing more organizations to collect, store, and process users' online information. The lowered cost barriers along with the development of data mining algorithms has made users' online information a valuable commodity to businesses. Companies can use the information to increase revenue by generating personalized product selections, performing targeted advertising, predicting user trends, etc. The more personal information that can be obtained about a user, the more accurate and effective these marketing methods can be. Businesses are motivated to collect and process as much data as possible, at the expense of user privacy. An additional factor is the ubiquity of the Internet. Organizations are able to collect personal information about users anywhere in the world, at virtually any time. This also means that organizations can place their resources in countries with little or no cyber laws, allowing them to circumvent online privacy laws of stricter countries.

1.2 Anonymous Peer-to-peer Protocols

Anonymous communication protocols allow users to communicate while hiding their physical locations. A few example uses of anonymous communication protocols are electronic voting, electronic cash, anonymous informants (whistle blowers), and censorship-resistant systems. All of the anonymous communication protocols that we have examined use a peer-to-peer architecture, so we refer to them as anonymous peer-to-peer (ANP2P) protocols. Currently the most popular ANP2P protocol

in widespread use is Tor [27], which enables users to anonymously browse the Internet. It should be noted that anonymous communication protocols can only provide an anonymous communication channel; anonymity of the content within messages is not provided. For example, if a user's personal information is included in a message, that information can be used to identify the user. Anonymous communication is not suitable for all applications; however, it is a desired system property in some situations. Detailed discussions on the different types of anonymous communication systems can be found in Section 2.2.

We classify ANP2P protocols into two groups based on their functionality types: anonymous access and anonymous data sharing. All ANP2P protocols support anonymous overlay networks. The bottom layer (the underlay network) provides basic communication functionalities (e.g. Internet). The topmost layer (the overlay network) builds on the underlay network and adds anonymity properties. *Anonymous access* protocols provide users with anonymous communications to resources that are located in the underlay network. Tor is an example of an anonymous access overlay network. When a user accesses a website via Tor, the request will first be routed through the Tor overlay network. This provides the user with a certain degree of anonymity. Then the request drops out of the overlay network and is routed the rest of the way through the underlay network (Internet) to the destination web server. *Anonymous data sharing* protocols provide users with anonymous communications to resources that are located in the same network overlay. Freenet [17] is an example of an anonymous data sharing protocol. When a user requests a document, the request is routed entirely through the overlay network. The key difference between the two categories of ANP2P protocols is that in the anonymous data sharing the message never leaves the overlay network in order to reach its destination. Refer to Section 2.2.1 for a more detailed discussion and figures about the anonymous access and anonymous data sharing categories.

We focus our study on ANP2P anonymous data sharing protocols. Since anonymous data sharing protocols implement their own destination-based routing in the overlay network, they can use anonymous routing techniques that differ from the traditional routing models used by the Internet. This gives anonymous data sharing protocols more control over how messages are routed from a source to a destination. The extra control gives the protocols more options for improving the anonymity provided to users. Additionally, several ANP2P anonymous access protocols have already been heavily studied, so we focus on the ANP2P anonymous data sharing protocols.

There are four basic types of ANP2P anonymous data sharing protocols: flooding, DC-net, rendezvous, and DHT. *Flooding* is the simplest type of protocol. It will forward a message to all of the nodes in the network in order to reach the destination node. Sending a message to every node in the network generates a lot of overhead traffic, and it causes scalability issues. *DC-net* (Dining Cryptographer network) ANP2P protocols use multi-party computations to provide provable anonymity. However, they are impractical for most real-world applications because they have a large message overhead. *Rendezvous* points are typically used by anonymous access protocols

to extend their functionality and make them an anonymous data sharing protocol. A directory service is required to publish the rendezvous points, so senders can look up destination nodes to find their rendezvous points. The last type of ANP2P anonymous data sharing protocol is *DHT* (distributed hash table). DHT protocols organize nodes and information into a hash table, using either semi-structured or structured networks. This organization makes the networks more efficient and scalable. Refer to Section 2.2.2 for a complete discussion on the different types of routing in ANP2P protocols.

We focus our study on ANP2P DHT protocols, as they provide the most desirable properties among the anonymous data sharing protocols. DHT protocols are scalable and efficient, and they can be fully distributed with little or no centralized control. The flooding and DC-net protocols suffer from scalability and inefficiency issues that make developing large scale networks a challenge. The rendezvous protocols require a public directory, and the directory is susceptible to harvesting, DoS, and scalability issues.

There are several open research issues with ANP2P DHT protocols. First, it is difficult to quantify the anonymity provided by most ANP2P protocols, with the exception of DC-net protocols. Since it is difficult to quantify the anonymity, we do not have an effective method to provide users with accurate anonymity guarantees. Second, there is often a trade-off between anonymity and performance that is not well understood. Performance is an important property required to attract and retain users; however, performance-improving mechanisms often decrease the anonymity provided by an ANP2P DHT protocol. Third, ANP2P DHT protocols are vulnerable to the Sybil attack, which allows an adversary to own a disproportionate number of nodes in an ANP2P DHT system. Finally, there are several open issues related to secure routing in ANP2P DHT protocols, such as the secure assignment of node identities, secure message forwarding, and secure peer selection. Refer to Section 2.5 for a more detailed discussion of the open research issues with ANP2P DHT protocols.

1.3 Contributions

The objective of our study was to better understand the fundamental design issues in ANP2P DHT protocols. We wanted to study how different design decisions affect the anonymity provided by the protocols. We also wanted to study how anonymity affects other properties of ANP2P DHT protocols. We were most interested in the relationship between anonymity and performance, and how one impacts the other. We developed a methodology that can be used to evaluate the performance and anonymity of generic ANP2P DHT design decisions. We used this methodology to measure and better understand how different design decisions impact the anonymity and other properties of ANP2P DHT protocols.

We performed a protocol analysis of Freenet and GNUnet, and we identified three vulnerabilities that could be used to compromise anonymity. The first was the *Traceback attack*, which exploits

protocol behavior and the order of operations to successfully determine if a node has previously routed a given message. The result of the Traceback attack is a set of potential sender nodes. Our colleagues Tian and Duan then formalized the attack, proposed methods for further reducing the sender set, and evaluated their methods in [80].

The second vulnerability we developed was the *Routing Table Insertion (RTI) attack*. This attack was created to help enable the Traceback attack. A requirement of the Traceback attack is that the adversary has a direct peer connection with the node that they want to test. The RTI attack abuses peer replacement policies in Freenet to help an adversary force itself into a victim's routing table. We then evaluated the impact of the attack and proposed two solutions. First, randomized routing can prevent adversaries from predicting routing paths. We evaluated the performance and anonymity impact this change had on the protocol. The second mitigation to the RTI attack is *Look Ahead Hint*. We proposed a variation of the original look ahead mechanism used by ANP2P DHT protocols to improve performance. Look Ahead Hint only shares a masked node identifier, in which only the high order digits are shared. We then evaluated the performance and anonymity impacts of using Look Ahead Hint.

The last vulnerability that we identified during our analysis is related to the bloom filter GNUnet uses for loop detection. Each message sent in GNUnet includes a bloom filter that contains an entry for every node that routed the message. We theorized that the bloom filter can be used to perform an attack very similar to the Traceback attack.

After analyzing the protocols, we proposed an empirical methodology that can be used to evaluate the performance and anonymity of ANP2P DHT design decisions. The methodology was developed to better understand the anonymity trade-offs within design protocols. Before developing this, we found it very difficult to quantify the anonymity provided by the counter measures proposed for the RTI attack. The methodology we developed uses a novel generic adversarial model that uses a routing path Walk-back Ranking Algorithm along with shared states and protocol behaviors to sample the entropy in a DHT protocol. The empirical methodology was run on a subset of the design decisions that we have encountered in ANP2P DHT protocols. The routing performance and observed anonymity metrics for topology type and look ahead were evaluated. Those metrics allowed us to compare various design choices and their impact on performance and anonymity.

1.4 Organization

In this chapter we have briefly discussed what ANP2P protocols are, and categorized ANP2P protocols into two categories: anonymous access and anonymous data sharing. We focused our research on anonymous data sharing protocols, of which there are four different types: flooding, DC-net, rendezvous, and DHT. Within these, we focused our research on DHT protocols because they are better suited for large scale applications. Next, we briefly listed some of the open research issues with ANP2P DHT protocols. Finally, we discussed the issues that we are going to address

in our study.

The remainder of this dissertation is organized as follows. In Chapter 2, we will discuss the different kinds of ANP2P protocols and how they work. We will also discuss current anonymity metrics, anonymity analyses that have been performed on ANP2P protocols, and open research issues with ANP2P DHT protocols. The list of assumptions we made about our analysis can be found in Chapter 3. In Chapter 4, we will perform an analysis of the Freenet and GNUnet protocols for vulnerabilities that could compromise anonymity. We will catalog the observed design choices in Chapter 5. Then, Chapter 6 will describe the empirical methodology we have developed to evaluate the performance and anonymity. We will apply the methodology to a subset of the design choices. The conclusion of our study will be presented in Chapter 7.

CHAPTER 2
BACKGROUND AND RELATED WORK

This chapter covers related research and relevant background information. First, a list of common terms that are used throughout this study are defined. Second, existing ANP2P protocols and their histories are cataloged. Next, research related to anonymity metrics is reviewed. Then, we look at several existing anonymity analyses that have been performed on ANP2P *anonymous data sharing* protocols (for anonymous data sharing definition see Section 2.2.1). Finally, the open research issues related to ANP2P anonymous data sharing protocols are discussed.

2.1 Anonymity Definitions

We formally define several terms that are used in this dissertation.

- **Anonymity**: The quality or state of not being identified or named. Pfitsmann and Waidner [62] identified three anonymous communication properties: *sender anonymity*, *receiver anonymity*, and *unlinkability of sender and receiver*. Anonymity protocols attempt to provide one or more of these communication properties. A sender is defined as the entity that initiates communication with a receiver. A receiver may choose to reply to a sender's communication if replies are supported by the anonymous protocol.

  1. **Sender Anonymity**: The identity of the entity who sends a message is hidden.
  2. **Receiver Anonymity**: The identity of the entity who receives a message is hidden.
  3. **Unlinkability of Sender and Receiver**: If a sender and receiver can be identified as participating in some communication, then they cannot be identified as communicating with each other.

- **Degree of Anonymity**: Most anonymous systems cannot guarantee anonymity; instead, they provide a probability of anonymity given a set of circumstances. Rieter and Rubin [67] informally defined a continuum to categorize anonymity provided by a system. The continuum has six categories ranging from *absolute privacy* on the high end to *provably exposed* on the low end. We are only going to further define three of the intermediate states because these are the only achievable anonymity states. The anonymity categories are defined in terms of *sender anonymity*, but they can be applied to *receiver anonymity* as well.

  1. **Beyond Suspicion**: If the attacker can see evidence of a sent message, the sender appears no more likely to be the originator of that message than any other potential sender in the system. This is the highest level of achievable anonymity.

2. **Probable Innocence**: If the attacker can see evidence of a sent message, the sender appears no more likely to be the originator than to not be the originator. This is weaker than beyond suspicion in that the attacker may have reasons to expect that the sender is more likely to be responsible than any other potential senders, but it still appears at least as likely that the sender is not responsible.
3. **Possible Innocence**: If the attacker can see evidence of a sent message, there is a non-trivial probability that the sender is someone other than the actual originator. This is the lowest level of achievable anonymity.

- **Anonymity Set**: If a given set of entities are considered indistinguishable from each other, then that set is considered an anonymity set for any given entity in it. The larger the anonymity set is, the stronger the anonymity it provides. Most anonymous protocols use the concept of anonymity sets to achieve anonymity.

- **Peer-To-Peer**: (P2P) is a decentralized and distributed network architecture. Each node in a P2P architecture can act as both a supplier and consumer of resources. P2P systems often have little or no centralized control. A node is also referred to as a peer.

- **Node**: In the context of this dissertation, a node has two meanings. 1) As previously described, a node is an entity participating in a P2P architecture. 2) A node is an instance of anonymous software running on a user's computer or mobile device that allows a user to access the anonymous system. When we use the term "node", it is in the context of ANP2P protocols and both definitions of a node apply. A node is also referred to as a peer.

2.2 Anonymous Peer-to-peer Protocols

We briefly discuss the history and influential work that has shaped the development of ANP2P protocols. First, we provide a high level overview of ANP2P protocols. Then, we define two categories for classifying ANP2P protocols: functionality type and routing type. It should be noted that the two categories are not independent of each other, and an ANP2P protocol's routing type directly affects the functionality it can provide. Finally, a brief history of the progression of ANP2P protocols is discussed.

When discussing ANP2P protocols we treat them as overlay networks. An overlay network is a computer network that is built on top of another network (e.g., IP network). Figure 2.1 illustrates an example ANP2P overlay network that is built on top of an underlay network. The connections between the nodes in the overlay network are virtual or logical links, meaning that a single logical link in the overlay network may be comprised of several physical links in the underlay network. The benefit of treating ANP2P protocols as overlay networks is that it can allow the protocol to

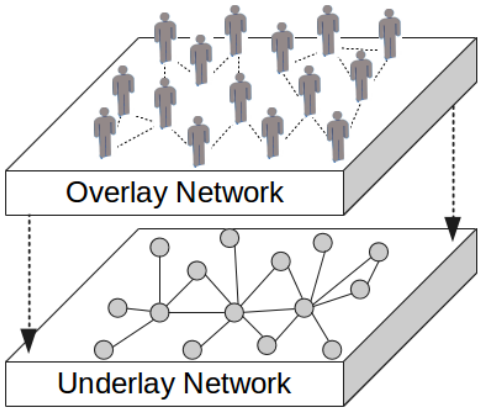

Figure 2.1: Anonymous peer-to-peer overlay network.

abstract away many of the details of the underlay network and how it functions. This allows us to focus more on the ANP2P protocol.

2.2.1 Anonymous Peer-to-peer Protocol Categories

The first category defines what functionality the ANP2P protocol supports. We have identified two general types of functionality: *anonymous access* and *anonymous data sharing*.

Anonymous access protocols use an ANP2P overlay network to provide users with anonymous communication access to resources located in the underlay network. Figure 2.2 illustrates an example message path through an anonymous access protocol. First the message is routed through several nodes within the ANP2P network. Only the nodes A, B, and C are part of the ANP2P overlay network. Then the message will leave the overlay network and continue routing to the final destination through the underlay network. The underlay network in anonymous access protocols is used for message transport and the resources it provides to users. Typically, anonymous access systems route through a series of nodes in the ANP2P overlay. Then the last node acts as a proxy for the original sender and sends the message directly to the resource located in the underlay network. An example anonymous access system is Tor [27]. It uses the Internet (TCP/IP) as the underlay network and provides access to resources in the underlay network (e.g., web browsing). A benefit of using an anonymous access system is that the ANP2P overlay network can use the routing functionalities of the underlay network. This reduces the number of functionalities the ANP2P overlay network needs to implement. Networking functionalities from the underlay network can be used to supplement network functionalities in the ANP2P overlay.

Anonymous data sharing protocols use an ANP2P overlay network and all messages are routed within the overlay network. This means that the underlay network is only being used to provide the node-to-node virtual links, and the underlay network doesn't directly contribute in

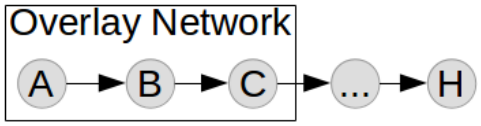

Figure 2.2: Anonymous access overlay network.

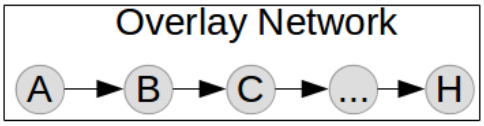

Figure 2.3: Anonymous data sharing overlay network.

routing a message from its source node to the destination node. Figure 2.3 illustrates an example message path through an anonymous data sharing protocol. The message is routed through a series of nodes that are all part of the ANP2P overlay network. Typically, these protocols are used to share digital content with other users in the ANP2P system (hence the name *anonymous data sharing*). The underlay network in anonymous data sharing protocols is only used to transport messages between overlay nodes through their virtual or logical connections. Freenet [17] is an example anonymous data sharing protocol. All senders and receivers are nodes in the Freenet network. The underlay network (Internet) is only used for transport purposes between nodes, and it is unable to read any messages it transports. Anonymous data sharing protocols need to implement all of the network functionalities that are required to route a message from point a to point b.

The second category defines how anonymity is achieved in ANP2P protocols. We divided the ANP2P protocols up into seven general types: *mix networks*, *DC-net*, *onion routing*, *random walk*, *flooding*, *split message*, and *distributed hash table (DHT)*. In Section 2.2.2, the basic design of each of these system types is discussed and then further categorized as anonymous access or anonymous data sharing. We also discuss the influential protocols for each of the ANP2P types.

2.2.2 History of Anonymous Peer-to-peer Protocols

The timeline in Figure 2.4 provides a sample overview of the research efforts that have been used for ANP2P protocols. The timeline depicts the initial development dates for several well-known and successful ANP2P protocols. The figure also categorizes the ANP2P protocols as either anonymous access or anonymous data sharing.

The **mix network** was published by Chaum [15] in 1981, and it was the seminal work on anonymous communications. The mix network is an anonymous email system that provides sender anonymity. A mix network uses one or more mix servers. These mix servers act as proxy servers that pool senders' messages. Then when enough messages have been pooled, they are shuffled and

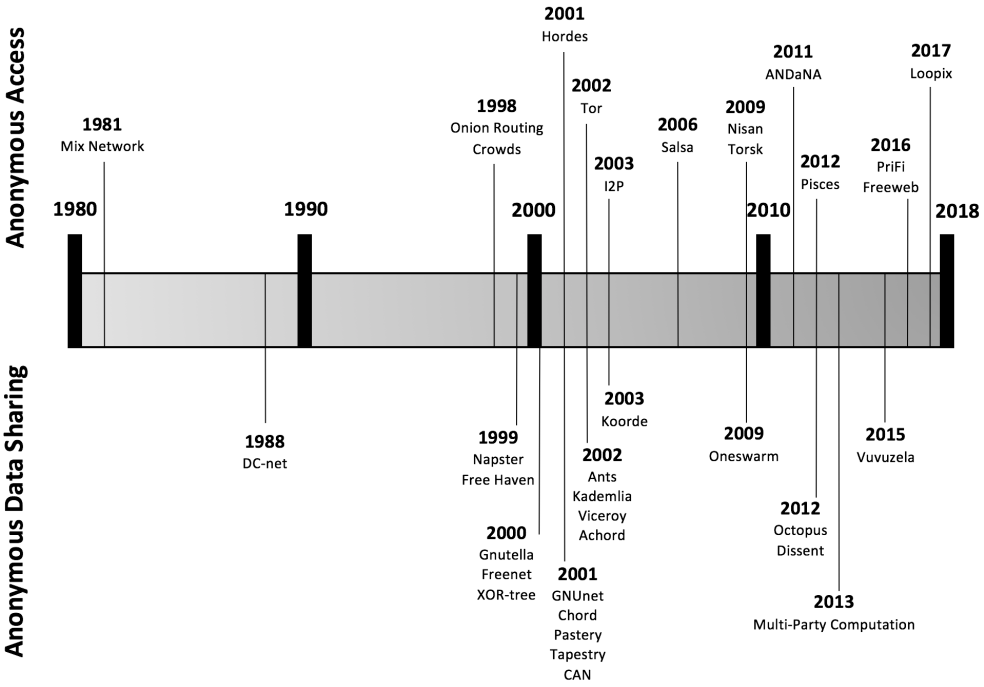

Figure 2.4: Time line of creation dates for a sample of ANP2P protocols.

sent out in a random order. This prevents observing attackers from correlating messages going into a mix server with the messages coming out. Because mix servers pool senders' messages, a mix network can only be used for delay-insensitive applications such as email. A mix is considered to be a basic anonymity design principle, and most ANP2P protocols use some variation of a mix. An example of a mix network is anonymous remailers. There are three types of anonymous remailers that have evolved over time: type I cypherpunk [61], II mixmaster [55], III mixminion [21]. Cypherpunk remailers take an email as input. The email can be encrypted with the remailer's public key or not. The remailer will then remove the sender information and send the email to the recipient. Recipients are unable to respond to messages unless the user included a reply address in the message body. Mixmaster remailers use a client side application to enable anonymous communication between the users and the remailer. In Mixmaster, like Cypherpunk, the messages are one way, and a reply address must be supplied in the message body. Mixminion remailers attempted to address several challenges in the Mixmaster design. Mixminion remailers provided features like message replies and replay prevention.

Mix networks are used to provide anonymous access. Vuvuzela [84] is a scalable mix network that uses cover traffic and dead drops. The protocol does not support low latency applications but claims protection against global observers. Vuvuzela uses dead drops to communicate between senders and receivers. A random Vuvuzela server is selected and used as the dead drop location.

The sender deposits the message at the agreed dead drop, and then the receiver reads the message from the dead drop location. Vuvuzela also uses cover traffic to prevent traffic analysis, and a fixed chain of servers for the mix network. Loopix [63] is a low latency mix network that uses cover traffic and Poission mixing. Loopix has an overhead delay of roughly 1.5 ms, and it claims to be resistant to global adversaries.

The Dining Cryptographers protocol (**DC-net**) was published in 1988 by Chaum [14]. DC-net is one of the few ANP2P protocols that has provable anonymity. However, DC-nets are impractical for most applications because of the large communication overhead they incur. Assuming N nodes in a group, then $\mathcal{O}(N^2)$ messages must be exchanged for each message sent. A small N can be used to limit the message overhead; however, that will reduce the size of the anonymity set, which reduces the overall anonymity provided by the protocol. Dolev and Ostrobsky [28] published a system similar to DC-net that uses XOR-trees. A multi-party computation algorithm proposed in [91] reduces communication overhead from $\mathcal{O}(N^2)$ to $\widetilde{\mathcal{O}}(N)$ per message sent. The overhead reduction comes at the cost of increasing latency from $\mathcal{O}(1)$ to $polylog(n)$. Dissent [88] is a DC-Net based protocol that uses a client-server architecture. The client-server architecture is used to reduce overhead and increase scalability of the protocol. Dissent only requires that the user trusts at least one of the servers. The authors were able to scale Dissent to 5000 servers. PriFi [1] applies the Dissent concepts to WiFi routing in a local area to provide anonymous access. DC-net based protocols are generally used to provide anonymous data sharing.

The **onion routing** protocol was published in 1998 by Reed, Syverson, and Goldschlag [66]. Onion routing is similar to Chaum's mix cascade (chain of mix servers) [15]; however, it provides low-latency communication. Onion routing provides sender anonymity. Messages are routed using Core Onion Routers (CORs). A sender will randomly pick a set of CORs to route through. It is assumed that the sender has access to the public keys for all of the CORs. A message is sent in an onion, which is a recursively layered data structure that contains information that CORs need for routing. When a COR receives an onion, it peels (decrypts) the outer layer with its private key. The decrypted information includes the routing information for the next hop, which is either another COR or an exit point. If it is an exit point, then the address from the message's final destination is given. An exit point will act as a proxy for the sender. Replies are sent back to sender in a reverse order through the CORs. Each COR will add an additional layer of encryption before forwarding a reply message. The sender can then remove all of the layers of encryption on the reply message. Each COR only knows about the previous and next COR, and a COR cannot read the message being sent to the next COR. There are several ANP2P protocols based on the onion routing protocol: Web Mixes [10], Tor [27], I2P [44], and Salsa [58]. Basic onion routing functionality provides anonymous access. However, rendezvous points can be added to allow anonymous data sharing. Tor and I2P are example protocols that can utilize rendezvous points.

Crowds is a **random walk** protocol that was published in 1998 by Reiter and Rubin [67]. Crowds provides sender anonymity by routing all messages with a random walk. When a node (called a jondo) receives a message, a weighted coin is flipped with two possible outcomes. The first outcome is that the node will continue to forward that message to another random node. The second outcome is that the node will act as a proxy and deliver the message to the final destination. Each message has a unique identifier that allows reply messages to be routed back to the sender through the same route taken to the receiver in reverse. Shortcut [89], AP3 [53], and Hordes [48] are all based on the random walk protocol. Random walk based protocols are used to provide anonymous access.

Flooding ANP2P protocols send messages to all nodes in the system to perform operations. The first popular flooding-based P2P system was Gnutella in 2000. When a user searches for content in Gnutella, the search request is sent to all of the peers directly connected to a node. Then each of those peers recursively forwards the search request to all of their peers until a time-to-live (TTL) counter is used up on the messages. This means a message is flooded to all nodes within the TTL range of the sender. Replies are routed back using the same path taken to a receiver. Scalability is an issue with flooding based systems, and each message sent generates a large amount of communication overhead. Message broadcasting and multi-casting are two variations of the flooding mechanism that can be used. Gnutella was not designed to provide anonymity, but it does provide some sender anonymity. It only provides weak protection, and there are several attacks [92] that can be used to identify the sender of a message. Ants [37], Oneswarm [43], and Hordes [48] all use flooding-based routing, and they are designed to provide anonymity. Hordes actually uses random walk routing to a receiver, and then the receiver uses a multi-cast reply back to the sender. Flooding-based ANP2P protocols can provide both sender anonymity and receiver anonymity, and they are typically used to provide anonymous data sharing.

Split message ANP2P protocols divide content up into several pieces that cannot be read individually. Several of the pieces must be obtained before the original content can be reconstructed. Free Haven is a split message system that was published by Dingledine, Freedman, and Molnar [26] in 1999. It splits content up into n shares. A share on its own is not readable; however, if a user can collect at least k of the n shares, then the original content can be reconstructed. Free Haven uses the Rabins information dispersal algorithm to split content up into n shares. The Free Haven network is made up of a number of servers known as servnets, which store and provide documents to anyone. Searches are flooded to all servnets. The sender can then combine the search results together to reconstruct the original content. Other split message systems are SSMP [41], Puzzle [40], Rumor Rider a.k.a. RR [39], and Publius [85]. Split message ANP2P protocols provide sender anonymity and receiver anonymity. They also provide anonymous data sharing.

The last type of ANP2P protocols use a **DHT** (distributed hash table). In a DHT, each node is assigned an identifier (node-id) within some given range (e.g., a real number between 0 and 1).

The node-id is then used to determine a node's location in the ANP2P overlay. All data inserted into the DHT is also assigned an identifier (data-id) that is within the same range as the node-id. Typically hash functions are used to calculate the node-id and data-id, and the hash functions are known to all of the nodes in the system. This means that given the same input (e.g., text document) every node in the system will calculate the same identifier. When data is inserted or requested from a DHT, the node that has the closest node-id to the given data's data-id will be responsible for that data. ANP2P DHT protocols provide sender anonymity. They also provide anonymous data sharing. There are two types of ANP2P DHT protocols: unstructured and structured.

Unstructured DHT ANP2P protocols do not have any guarantees about the structure of the P2P overlay. Unstructured protocols also include semi-structured protocols, which use best effort to provide a structure constant. Freenet [19] is a popular semi-structured ANP2P DHT protocol that was published in 2000 by Clarke, Sandberg, Wiley, and Hong. It uses a semi-structured DHT that tries to maintain a small world topology [71]. Freenet provides censorship-resistant document storage and retrieval. GNUnet [8] is another ANP2P DHT protocol.

Structured DHT ANP2P protocols maintain some constant (mathematical invariant) in regard to their topology structures. Typically, this invariant is used to guarantee performance in the protocol. There were four structured ANP2P protocols published in 2001: Chords [78], Pastry [69], Tapestry [93], and CAN [65]. Each of these maintain a different topology invariant; however, most of them can achieve a routing performance of roughly $\mathcal{O}(\log n)$. Kademlia [51], Viceroy [50], Koorde [45], Achord [42] are several other structured ANP2P DHT protocols. Freeweb [75] uses a combination of structured DHT and onion routing. A structured DHT is used to build circuit paths to exit proxies. Then onion routing is used to construct the reply circuit from the exit proxy back to the sender. The reply onion routing path is built using entries in the sender's routing table. Onion routing is used for the reply circuit because the DHT routing is not deterministic. DHT routing will continue until a node that can process the request is found. Freeweb provides anonymous access.

2.3 Anonymity Metrics

Several anonymity metrics have been proposed for ANP2P protocols. These metrics can be used to determine the *degree of anonymity* provided by an ANP2P protocol. The basic idea behind these metrics is to measure how distinguishable a node is within an anonymity set. The notion of the anonymity set is introduced in [14]. The size of an anonymity set is used to quantify the system's provided anonymity. We briefly cover three different types of anonymity metrics: process-calculi, probability, and information theory.

2.3.1 Process-calculi

Communicating Sequential Processes (CSP) is a formal language that can be used to define interactions in concurrent systems. CSP was used to formally define anonymity properties in [72] and [70]. CSP was applied to a DC-net example, and its anonymity was evaluated using the FDR model-checking tool.

2.3.2 Probability-Based Methods

Probability-based metrics are the most popular metrics used for evaluating the potential anonymity provided by an ANP2P protocol. In 2004, Shmatikov [76] used the PRISM probability model checker to analyze protocol anonymity. The nodes in an ANP2P protocol were modeled using a discrete-time Markov chain. This captured the behavior of honest and malicious nodes. The desired anonymity properties of the system were then expressed as probabilistic computation tree logic (PCTL) formulas. PRISM was then used to evaluate the anonymity properties. Using this method, Crowds was modeled when it was under attack by collaborating malicious nodes that were trying to find the origin of a message. This method is limited by the state space of the problem, and it can only model a small number of nodes in an ANP2P protocol.

Guan, Fu, Bettati, and Zhao [36] used a probabilistic method to determine the probability of an attacker discovering the sender of a message in 2004. They assumed that there were a given number of corrupted nodes in the networks, and that those nodes could appear anywhere in the routing path. Various ANP2P routing path selection strategies were analyzed. They considered path length, path complexity, and the number of corrupt nodes present, and how these properties affected anonymity.

In 2008, Murdoch and Watson [57] used the probability of a path being compromised as a measure of anonymity of Tor. They analyzed how the anonymity of different path selection algorithms were affected by the presence of malicious nodes. The attack model was slightly different from most models. They assumed bot-nets could be used to perform attacks against Tor. As a result, the attacker had access to a large number of nodes with low bandwidth and diverse IP addresses.

Feigenbaum, Johnson, and Syverson [34] performed a probabilistic analysis of onion routing in 2012. They assumed that attackers were active, controlled a portion of the network, and had some a priori knowledge about user's probability distribution for selecting destinations. Their model took into account probabilistic user behavior and protocol operations.

2.3.3 Information Theory

There are several anonymity metrics that are based on information theory principles. Most ANP2P protocols use a random primitive to provide anonymity. These protocols work under the assumption that the attacker will be unable to determine the outcome of probabilistic choices in the system.

The probabilistic nature of information theory works well for evaluating most ANP2P protocols.

Entropy was used to measure anonymity in [74, 25]. *Entropy* measures the amount of uncertainty about the anonymous events in an ANP2P protocol. The higher the entropy value is, the less certain we are about the anonymous events, and the greater anonymity that is provided. If entropy is zero, then the system is completely deterministic and provides no anonymity. This metric requires that a probability distribution X is supplied. X contains probabilities p_i of each node being the sender of a message (assuming we are looking at sender anonymity). The sum of values in X will equal one. The amount of entropy in a system will change based on the probability distribution X that is given. The maximum entropy (H_M) a system can achieve can be calculated when all of the probabilities in X are equal. The function D (Equation 2.1) is the normalized entropy of the system after observing an anonymous event and N is the total number of nodes in a system. The functions D, $H(X)$, and H_M are from the work in [25].

$$D = \frac{H(X)}{H_M} \tag{2.1}$$

$$H(X) = -\sum_{i=1}^{N} p_i log_2(p_i) \tag{2.2}$$

$$H_M(X) = log_2(N) \tag{2.3}$$

In 2005, Zhu and Bettati [94] created an anonymity metric based on mutual information. Mutual information measures the amount of information that one anonymous event contains about another anonymous event. First, entropy is used to measure the uncertainty of anonymous events before the protocol executes. Then the protocol is executed, which produces a set of publicly observable events that an attacker can collect. The observable events can then be used to calculate the conditional entropy, which gives the uncertainty of anonymous events after an attacker has observed some events in the protocol. Finally, we can compare the entropy before protocol execution and the conditional entropy after to obtain the mutual information. Mutual information measures the information about the anonymous events that were contained in the observable events. The mutual information metric requires that the probability distribution X of user probabilities is supplied, as well as the set of observable events. The function $I(X;Y)$ measures the mutual information of the random variables X and Y.

$$I(X;Y) = H(X) - H(X|Y) \tag{2.4}$$

Chatzikokolakis, Palamidessi, and Panangaden [13] used channel capacity to measure the maximum potential anonymity provided by a protocol in 2008. Channel capacity is an abstraction of mutual information obtained by maximizing over the possible input probability distributions. By maximizing the input probability distributions (X), X does not need to be supplied to calculate the channel capacity. Only the set of observable events is then needed. Chen and Malacaria [16]

extended the channel capacity metric using the Lagrange multiplier to remove the assumption that the protocol is symmetric in 2009. This allows the metric to be used for most anonymous protocols.

In 2010, Chatzikokolakis, Chothia, and Guha [12] developed a methodology that can calculate the mutual information and channel capacity metric from trial runs of an anonymous protocol. They have developed a tool to automatically perform these anonymity metric calculations on a mix network simulation.

In 2009, Syverson [79] analyzed the effectiveness of entropy-based metrics. Syverson stated several examples where entropy-based metrics do not capture the actual anonymity provided by a protocol, and that entropy-based metrics are not a complete metric of anonymity. In conclusion, it is suggested that entropy-based metrics should not be the only metrics used to evaluate the anonymity of a protocol.

2.4 Anonymity Analysis of ANP2P Data Sharing Protocols

In this section, we will discuss additional related work on evaluating anonymity in ANP2P data sharing protocols. The protocol specifications in Section 2.2.2 typically provide some initial anonymity analyses to justify the design decisions in the protocol. The anonymity metrics in Section 2.3 are typically applied to an example ANP2P protocol to justify the metric and evaluate anonymity provided by that protocol. We then discuss additional work that evaluates anonymity in ANP2P protocols. Our work focuses on data sharing protocols, so we limit the scope of anonymity analysis work to ANP2P data sharing protocols.

There are two main unstructured DHT ANP2P protocols in wide spread use: Freenet [17, 19] and GNUnet [9, 8]. McCoy [52] performed an empirical study on anonymity provided by Freenet in 2004. A series of passive attacks were performed on the Freenet network and analyzed to determine the effectiveness of the attacks. McCoy suggested adding latency to counter the passive attacks that were performed. In 2005, Borisov [11] measured the anonymity provided by Freenet using the entropy metric [74, 25]. Several weaknesses were identified in Freenet's design that could compromise anonymity. Specifically, high degree nodes were identified as being vulnerable to attack. In 2014, Roos, Schiller, Hacker, and Strufe [68] investigated Freenet performance issues by analyzing the topology. Several user behaviors were recorded and analyzed: number of active users, churn behavior, and file popularity. In 2017, Levine, Liberatore, Lynn, and Wright [47] used a statistical attack to determine if the previous node in a routing path was the original sender or a forwarding router. The attack only required a single passive peer to carry out, and it achieved a false positive rate of about 2% in experiments. The attack worked on the assumption that large files were split up into smaller and more manageable blocks, and that the original sender would evenly distribute its file block requests among all its peers. Based on the number of block requests the attacking peers saw out of the total number of blocks that made up a file, the attacking peers could statistically determine if the previous node was the original sender or a forwarding router.

In 2003, Kugler [46] performed an anonymity analysis on GNUnet. They showed that performance improving mechanisms in GNUnet could be exploited to determine the initiator of a download.

The next group of anonymity analysis work is related to structured DHT protocols. In 2002, Hazel and Wiley [42] analyzed the anonymity provided by Chord [78], and then presented Achord, a variation of the Chord protocol. Achord was designed to address the issue of receiver anonymity in Chord. Condie *et al.* [20] examined the impact of routing table poisoning attacks against structured DHT ANP2P protocols in 2006. They presented an induced churn method to mitigate routing table poisoning attacks. In 2007, Baumgart and Mies [7] analyzed the anonymity provided by Kademlia [51], and then presented a variation of the protocol called S/Kademlia [7]. The most interesting contribution from S/Kademlia was the use of crypto-puzzles in an attempt to make the Sybil attack computationally infeasible. Mittal and Borisov [54] studied the information leaked in look-up mechanisms used in structured DHT ANP2P protocols in 2008. The protocols Salsa [58] and AP3 [53] were investigated. They found that using a combination of active and passive attacks could compromise anonymity. In 2011, Urdaneta, Pierre, and Steen [83] provided a survey of structured DHT security issues.

The last group of work is related to flooding-based ANP2P protocols. In 2011, Prusty, Levine, and Liberatore [64] investigated the anonymity provided by Oneswarm [43]. They showed that Oneswarm was vulnerable to timing-based attacks and provided some figures on the resources attackers needed to compromise the network. The biggest challenge with flooding-based ANP2P protocols is scalability, and as a result most work is focused here.

2.5 Open Research Issues

There are several open research issues for ANP2P protocols. The issues that are relevant to our research will be briefly covered in the following section. We discuss the impacts of the open research issues and any existing proposed solutions or mitigations.

2.5.1 Secure Peer Selection

How a node selects its peers can drastically affect the anonymity provided by the protocol. In an ANP2P, a node's peers significantly affect how it communicates with others. These peers can either be directly connected to a given node and make up its routing table (e.g. in DHT-based systems), or they can be the peers selected to build a routing circuit (e.g. onion routing). In the best-case scenario, if an adversary controls some percentage f of all the nodes in the system, then on average any given node should only have an f chance of its peers being controlled by the adversary. For example, if 5% of nodes in the system are malicious, then only 5% of any individual node's peers should be malicious. However, if an adversary were able to exploit an ANP2P protocol's peer selection, then they could insert a disproportionate number of malicious nodes into a victim node's

peer list. This means that an adversary could attack an ANP2P protocol more efficiently with fewer resources.

It is difficult to determine which nodes are honest and which nodes are malicious, especially if the malicious nodes are only performing passive attacks. There have been several proposed solutions to counter this. First, when selecting peers, a node should take a diverse set of peers. For example, Tarzan [35] and some other ANP2P protocols require that only one peer can represent a subnet of IP addresses. The idea here is that if a single adversary wanted to take over all of a node's peers, then they would have to own a diverse set of IP addresses. However, bot-nets are increasing the diversity of IP addresses available to adversaries. Second, reputation systems can be used to determine a node's integrity. Many ANP2P systems have tried using reputation-based systems in one form or another. The problem with this solution is that adversaries can circumvent the systems, and in some cases actually make the situation worse. It is easy for a malicious node to change its identity if it gets a bad reputation. Malicious nodes can also falsely inflate their own reputation using other collaborating malicious nodes. Malicious nodes that only perform passive attacks may not even be detected by the reputation system. Another issue with using reputation systems in ANP2P protocols is that they can leak information. In 2014, Das, Borisov, Mittal, and Caesar [24] did some work on developing a reputation system for path construction in Tor. The third solution to avoid a disproportionate number of malicious peers is that a node can randomly select its peers. This approach works if an adversary cannot affect the set of peers that a node randomly chooses from. Fourth, a node can run in a darknet mode (a.k.a. friend-to-friend routing). In a darknet, peers are located using social network connections. We discuss the merits of darknet in more detail in Section 2.5.3.

2.5.2 Secure Message Forwarding

Secure message forwarding is the ability to actively remove malicious nodes from the system or avoid routing through them. In order to remove a malicious node, there must be a method to determine which nodes are malicious. Currently there is no reliable method of doing this in ANP2P systems. Using a centralized authorization server can vet some malicious nodes out; however, a centralized server counters the design goals of decentralized systems. This issue is directly related to secure peer selection (see 2.5.1).

2.5.3 Sybil Attack

The Sybil attack [29] allows an adversary to own a disproportionate number of nodes in an ANP2P protocol. In this attack, an adversary tries to acquire as many nodes in the ANP2P system as possible. A larger the number of nodes (even if they are randomly located in the topology) gives an attacker greater control over the network. This allows a small number of users with large computation power to gain disproportionate control of the system. This attack is extremely difficult

to stop in ANP2P protocols. This is especially true in protocols that support opennet, where any node can be peered to any other node.

There are no complete solutions to this attack. A centralized solution can be used to limit the rate at which nodes enter the system, and further filter based on IP locality. Darknets (friend-to-friend routing) can also be used to mitigate the damage done by the Sybil attack. A darknet will prevent the adversary from becoming a direct peer to most peers in the system and prevent them from compromising sender anonymity. Work has been done in [90, 22] that shows the effectiveness of using social networks to mitigate the Sybil attack. However, if the adversary is able to get even a few malicious nodes into the network, then they can connect to their own malicious nodes and still take over a large portion of the ANP2P systems node space. This will allow them to control a portion of the content in the system.

Opennet Vs. Darknet. *Opennet* is a networking mode that allows a node to be peered with any other nodes. This means that it is possible for a node to randomly connect to malicious peers that will try to diminish the anonymity of the node. All peers are considered untrustworthy in opennet mode. In the *darknet* or friend-to-friend mode, a node will only connect to nodes that they trust. The users behind a node's peers are manually vetted, and the node has an elevated level of trust in the peer. Darknets will form social networks where relationships outside of the ANP2P protocol are used to establish trust. ANP2P protocols typically fall into either opennet or darknet, and sometimes both.

Many researchers believe that opennet cannot be secured because of the open research issues discussed here, but it has not been proven yet. Darknets can be used to protect a user from many protocol-related attacks on anonymity; however, they are susceptible to social engineering attacks. Because of the elevated trust placed in darknet nodes, a compromised darknet peer can be very detrimental to a node's anonymity. A user must also have a large number of darknet peers in order to maintain performance (peers commonly only participate in the system for limited period of time). Using a mixed mode of opennet and darknet helps a node fill their peer list when there are not enough darknet peers available. It should also be noted that any anonymity improving mechanism that can be applied to opennet can also be applied to darknet, but the reverse is not true.

2.5.4 Performance

Performance is a desirable property in any P2P protocol, and it is also true for ANP2P protocols. Users expect a level of performance from any ANP2P system based on experiences with similar systems. This expected performance is typically defined in terms of seconds and fractions of a second. It is important to note that perceived performance can have a disproportionate impact on user perception of a system [87, 73].

The problem is that performance and anonymity seem to be competing properties in ANP2P

systems. Several researchers have looked at the impact of performance versus anonymity in specific circumstances. However, there has not been a general methodology developed for quantifying the trade-off between performance and anonymity. As an example, structured DHTs provide performance guarantees in ANP2P protocols; however, additional information is leaked during the operations of these protocols [54]. The performance improvements of structured DHTs may come at a cost of anonymity.

2.5.5 Secure Assignment of Node Identity

In DHT-based protocols, a node's identifier is used to determine its location in the topology and is used to make routing decisions. If an adversary is allowed to choose its own node identifier (node-id), it can control the placement of malicious nodes in the topology. Allowing attackers to freely choose their node-ids enables the eclipse attack [77]. This attack allows the adversary to put malicious nodes in key locations on the topology or concentrate malicious nodes in one area of the topology. Concentrating malicious nodes can allow the adversary to perform attacks that compromise anonymity in a subsection of the topology with fewer resources. Changing node-ids will also allow the adversary to dynamically move their resources around the topology.

There have been several proposed solutions to this issue; however, they do not completely solve the issue, or they present new vulnerabilities. 1) Use a trusted centralized authority server to assign secure node-ids, and other nodes only accept peers that received their node-ids from the trusted server. The problem with this solution is that it creates a single point of failure in the protocol. It also requires that all nodes trust the server, and there are scalability issues when centralized servers are used. 2) Use a set of peer nodes to compute and assign the node ID. The original protocol for Freenet attempted to use this method [18]; however, it was removed from the protocol. The problems with this approach are that a set of nodes used to generate the node-id can be malicious, and there is no way to enforce that a node uses the node-id it is assigned. 3) Base node-id on a verifiable property of a given node. The Salsa protocol [58] hashes a node's IP address to determine its node-id. This way all of a node's directly connected peers can verify that the node-id used matches the node's IP address. The idea is that IP addresses are a scarce resource and can be easily verified. There are a of couple issues with this method. The emergence of bot-nets has made obtaining diverse IP addresses much easier for adversaries and using a node's IP address for its node-id may reveal too much information. Since node-ids are secure hashes, an adversary cannot directly take a node-id and convert it back into an IP address. However, there are a finite number of public IPv4 addresses($\approx$ 3.7 billion), and it is not unreasonable to pre-compute the hashes of all of those addresses. If an adversary were able to obtain a node-id, then rainbow tables [59] could be used to perform crypt-analysis and obtain the IP address of the node. 4) Use crypto-puzzles to make generating node-ids computationally expensive. S/Kademlia [7] uses crypto-puzzles. However, [60] points out that the crypto-puzzles can be solved offline before joining

the network.

2.5.6 Provable Anonymity

Provable anonymity is a difficult challenge in distributed protocols. DC-net [14] and closely related protocols are currently the only ANP2P protocols that provide provable anonymity; however, because of scalability issues they can only support a small anonymity set. Studies have been done on other ANP2P protocols to try to determine the anonymity they provide. Most of these studies only provide best guesses about anonymity and are unable to guarantee the level of anonymity they provide. This presents a challenge when convincing users that a given ANP2P protocol provides more anonymity than another ANP2P protocol. There is still active research into finding methods to verify anonymity provided by ANP2P protocols.

CHAPTER 3
ANALYSIS ASSUMPTIONS

There are several general assumptions that we have made to specify the scope of our research. In this chapter, we discuss those assumptions and why we made them. These assumptions will apply to the experiments and analyses performed in Chapters 4 and 6.

3.1 Adversary

When evaluating anonymity of an ANP2P DHT protocol, the capacities of an adversary must be defined. The adversary model is defined based on their views of the ANP2P DHT protocol (*global* or *local*), their positions relative to the ANP2P DHT protocol (*external* or *internal*), the behavior of the adversary (*passive* or *active*), and if the adversary can *collude*. In this section, we will discuss each capability of the adversary and that capability's general impact on anonymity in ANP2P DHT protocols. Then this section will be concluded with a summary of the adversary capabilities that we assume for this work.

3.1.1 View

A *global* adversary has the capability to monitor all network traffic in a given network. Global adversaries are the most threatening to ANP2P DHT protocols. Because the adversary can monitor the traffic in and out of all nodes, they have the greatest capability to break anonymity. For example, all low-latency ANP2P DHT protocols are particularly vulnerable to timing-based traffic analysis attacks when global adversaries are present. Depending on the number and geographical distribution of the nodes in an ANP2P DHT network, a global adversary may require significant resources to monitor all traffic. Getting access to the resources required for global monitoring is a challenge for most adversaries.

A *local* adversary has the capability to only monitor a fraction of all nodes. Typically, the adversary can only monitor nodes that are close logically or geographically. Adversaries with local monitor capabilities are considered to be the most common type. Most low-latency ANP2P DHT protocols are only designed to counter adversaries that have local monitor capabilities limited to monitoring some fraction f of the total nodes. As the fraction f increases, the potential for the adversary to break the anonymity also increases.

3.1.2 Behavior

Passive adversaries will participate in the ANP2P DHT protocol and observe communications. Passive adversaries are sometimes also called eavesdroppers. In general, passive adversaries have less

potential to compromise anonymity when compared to active adversaries. However, it is extremely difficult for honest nodes to identify passive adversaries because they produce no directly observable indications in an ANP2P DHT protocol of malicious behavior.

Active adversaries will observe, add, delete, and modify messages in an ANP2P DHT protocol. As a result, they have a greater potential to compromise anonymity. In addition to compromising anonymity, active adversaries can also perform other malicious actions to disrupt functionality in ANP2P DHT protocols. Active adversaries may be easier to detect than passive adversaries. Deviating from the protocol behaviors may be observed by other nodes in the system and be interpreted as malicious behavior.

3.1.3 Position

An *external* adversary does not directly participate in the ANP2P DHT protocol. It can only monitor the traffic coming in and out of nodes in the ANP2P DHT protocol. Typically, external adversaries can only perform eavesdropping attacks. Since most ANP2P DHT protocols use some form of encryption when communicating between nodes, external adversaries can only observe the publicly decipherable data in the observed packets (e.g. source and destination IP addresses).

An *internal* adversary participates in the ANP2P DHT protocol. Typically, an internal adversary will take the form of a malicious node that joins the ANP2P DHT network. Since an internal adversary is part of the ANP2P DHT system, it has access to more information than an external adversary. Internal adversaries can also leverage their understanding of the ANP2P DHT protocol to compromise anonymity. An internal adversary can typically only monitor the peers that they are directly connected to.

3.1.4 Colluding

A single adversary can have multiple malicious nodes in an ANP2P DHT protocol. If the adversary can coordinate the observations and actions of those nodes, then they are considered to be *colluding*. Typically, an out-of-band communications channel will be used to coordinate the multiple malicious nodes. Using colluding nodes can greatly increase the effectiveness of an adversary and also enable more classes of attacks. Because of the nature of ANP2P DHT protocols, it is assumed that all adversaries have colluding capabilities.

3.1.5 Conclusion

We have made the follow assumptions about the adversary's capabilities for this work. Since ANP2P DHT-based protocols are low-latency communication systems, we assume that the adversary only has *local* monitoring capabilities. In general, low-latency anonymous communication protocols are not designed to resist *global* adversaries. We are also assuming that the adversary is *internal*,

meaning that the adversary will attempt to join and participate in the ANP2P DHT protocol. It is easy for an adversary to join most ANP2P DHT protocols, and the monitoring capabilities that are gained from joining the ANP2P DHT networks are typically far greater than an *external* adversary would have. Finally, we assume that the adversary has *colluding* capabilities.

3.2 Anonymity

3.2.1 Sender Anonymity

There are three anonymity properties that any anonymous protocol can have: sender anonymity, receiver anonymity, and unlinkability of sender and receiver [62]. In this work, we focus on sender anonymity. An ANP2P DHT protocol typically uses a deterministic-based routing approach, which provides very little receiver anonymity. In addition, intermediate nodes on a routing path can view the target destination of a message. To address this issue, most ANP2P protocols use a rendezvous-based approach. The sender will write a message to a given location in the topology, then the receiver will go to that given location to read the message. In this approach, the sender and receiver of the message are both actually considered senders. Providing sender anonymity in turn also provide the unlinkability of sender and receiver.

3.2.2 Anonymity Sources

The source of the anonymity properties in ANP2P DHT protocols can be grouped into two categories: *protocol state* and *protocol behavior*. We will further define each of these categories and discuss the assumptions we have made about the categories. These assumptions will have an impact on the overall anonymity of the ANP2P DHT protocols.

The *protocol state* is defined by the values and properties of the nodes participating in the protocol. For example, a given node will have an address associated with it, a list of nodes that it is peered with, and a random number generator. If you combine the state of every node in a given protocol, then you will have the state of the entire protocol. The protocol state can be further broken down into *internal state* and *shared state.* The *internal state* of a node consists of properties that are only known by that node. The *shared state* is the set of properties that are shared between nodes. From the previous example, the random number generator is an example of internal state, and the peer list with corresponding addresses are shared states. We assume that an adversary can learn the entire shared state of a protocol. However, depending on how much state is shared and with how many nodes, it can be difficult for an adversary to recreate the shared state of the entire protocol. Future work would be to investigate the impact of shared state and anonymity.

The *protocol behavior* is defined by the actions a node will take in response to other actions or states. The behavior can also be called the logic of the protocol. For example, this includes how a node will route a message and select its peers. We assume that the protocol behavior is available to

an adversary. This means the adversary either has the source code available, has reverse-engineered the protocol based on observations, or has decompiled or reverse-engineered the executable binaries.

In conclusion, we assume the adversary has access to the behavior of the protocol (i.e. logic) and the shared state (e.g. topology). The only information that an adversary does not have access to is the internal state of the individual nodes that they do not control.

CHAPTER 4
ANONYMOUS P2P DHT PROTOCOL SPECIFICATION ANALYSIS

To better understand the anonymity provided by ANP2P DHT protocols, a study of existing protocols and their design decisions is needed. We chose to study two protocols: Freenet [19, 17] and GNUnet [8, 9]. Most of the content in this chapter is focused on our analysis of the Freenet protocol. In this chapter, we will discuss the general methodology that we used to analyze both protocols. Then several findings from our analysis will be discussed along with some proposed mitigations.

4.1 Methodology

We used two general approaches to study Freenet and GNUnet. The first approach was to model the protocol specification based on available design literatures [19, 17, 8, 9]. We call this the design specification. This gave us the general concepts of the protocol, the anonymity goals, and features that the protocol was trying to achieve. However, the derived design specification lacked details. The next approach was then to look at the actual implementation of the protocol and derive the protocol specification from source code and the executable. We call this the implementation specification. Next, we will discuss the methodology used to derive the implementation specification in more detail.

Source code was used to construct the implementation specification. The source code for Freenet and GNUnet was reviewed to better understand the protocol interactions. Then the source code was instrumented and compiled to build an executable protocol. The instrumented executable was used to trace the protocol interactions. The collected information from the two previous steps was then used to construct UML interaction diagrams and state transition diagrams, so we could better study the impacts of the design decisions. An example of a state transition diagram for one of Freenet's content request protocols can be found in Appendix A.1.

After constructing the design and implementation specifications for the Freenet and GNUnet protocols, our next step was to analyze those specifications. We looked for information leaks from the normal operation of the protocol. Then we looked for methods of abusing the protocol to reveal additional information about users. Abusing the protocol included things such as forcing the protocol into an improper state or sending messages to invoke a specific response. This means adversaries could manipulate the protocol into improper states and exploit those states to reveal information.

4.2 Freenet Overview

Most of the work in this chapter relates to the Freenet protocol. We will provide a brief overview of how the Freenet protocol works before continuing on to our first protocol specification analysis finding in Section 4.4.

Freenet uses a distributed hash table system to store and retrieve information from its network. The routing algorithm that Freenet uses can be characterized as a steepest-ascent hill-climbing algorithm. Every node in the Freenet network has a small world view of the entire system, and it will use its limited information to make the best routing decisions that it can. There are several sources [19, 17] that describe the details of the routing algorithm. However, the route prediction model described in this dissertation uses the routing algorithm defined in the source code (version 0.7.5) [30], as it was the most current version of the routing algorithm at the time of this study. Version 0.7.5 has a few differences from the original design documents [19, 17] . The protocol was initially proposed in 2001 by Clarke, Sandberg, Wiley, and Hong. The general Freenet network configuration, storage, and retrieval mechanics will be discussed. Also, the anonymity contribution from the Freenet routing algorithm will be covered.

In a Freenet network, each node operates independently. A node is client software used to participate in the Freenet network. A node can have a direct connection with up to 40 other nodes. These connected nodes are considered to be direct peers. Each node in Freenet has a node location that is a real number in the range $[0, 1]$. Node location is also referred to as the node address or node id. The address space is circular, *i.e.*, the address 0 and 1 are equal. The node address is used to determine routing paths, and it serves as an indicator of when a node should store information. A node will use the node addresses of all the peer nodes within two hops when it makes routing decisions. This means that a node potentially knows the node locations of $\approx 40^2$ other nodes.

Every piece of data that is inserted into Freenet has an address associated with it. The data is hashed into a key, and then the key is converted to a real number in the range $[0, 1]$. This number is the data address for a given piece of information. The data address can then be used in combination with the node addresses of the nodes to route and store information. The routing algorithm will try to find the node with the closest node address to the data address for a given piece of information.

Every request that is generated in Freenet has a unique identifier called a UID. The UID is passed with the request so nodes can identify a request that they have already handled. Each request has a data address associated with it. If a node is unable to directly service a request, it will forward the request to the next peer. The next node will be a direct peer that has the closest node address to the given data address. Requests will be forwarded until the request can be serviced or the hops to live (HTL) reaches zero. The HTL will be decremented each time a request is forwarded. If a request can be serviced, the reply will be sent back down the path that it came from to the original sender. How the next node is chosen, how HTL works, and the differences

between insert and retrieval requests will be discussed in detail in the following.

Each Freenet node will look at all of its peers that are within two hops. Based on the node location of those peers it will choose a node that is one hop away as the next peer to route to. When a node is forwarding a request to the next peer, it will never forward the request back to the node that it received the request from. Also, Freenet will not route to nodes that have already been visited with the current request. Since an individual node does not know the full list of nodes already visited by a request, the request may initially be forwarded to an already visited node. When a node receives a request, it will check if it has already seen a request with the same UID, and if it has, it replies with a reject loop message.

A request is forwarded until it can be serviced by a node or its HTL reaches 0. HTLs are used to help limit the amount of resources used by a request. Initially the HTL starts out at 18 and is decremented each time that a request is forwarded to another peer node. Two exceptions to this decrement rule are when HTL is equal to the maximum HTL value (18) or equal to 1. There is a 50% chance that the HTL will not be decremented when the HTL is at the maximum value. There is a 75% chance that the HTL will not be decremented when the HTL is equal to 1.

There are two common kinds of requests in Freenet. Data retrieval requests are used to get data out of Freenet, and data insert requests are used to insert data into Freenet. The data retrieval request is the simpler of the two, and it will continue to search until a node can service the request or the HTL expires. If the HTL expires, then the request fails. A data insert request is actually comprised of two separate data insert sub-requests. When the HTL on a data insert request reaches the cacheable HTL threshold of 15 (maximum HTL - 3), a new data insert request will be generated. The new request will have all the same parameters, including data address and HTL. The only difference will be that the new request will have a different UID. Using a new UID causes the visited node set to be cleared. Resetting the visited node set is done to avoid local minimums when inserting. The data insert request will then continue until the HTL expires. A data insert request will permanently store the data being inserted into every node that it passes through where the HTL is cacheable.

Most of the anonymity in Freenet is provided through its routing algorithms. Each node only has a small view of the entire world. A node can only tell who the previous peer node was and the next peer node for a given request. Even though a node knows who the previous node was, it cannot tell if that node was the request originator. Since there is a 50% chance the maximum HTL value (18) will not be decremented, when a node gets a message with a value of 17 (maximum HTL - 1), that does not necessarily mean the previous node was the origin node. Also, since each node is operating independently without any central control, it is difficult to monitor all traffic in the Freenet network.

4.3 Design vs. Implementation

When we compared the design specification to the implementation specification of Freenet, we found several differences. Although it is often intuitive to make certain high-level design decisions to improve the anonymity strength of the consequent ANP2P DHT protocols, such as message aggregation or mix, it is much harder for developers to make the best low-level implementation decisions in practice. E.g., how to incorporate a new node into the network, how to share routing information among peers, how to use certain peers in proper ways without introducing fixed patterns that can be exploited by attackers, or how to limit such patterns from being exploited. This difficulty is due to a number of often-conflicting factors that must be considered in the low-level implementation process. Unfortunately, low-level implementation decisions are often not well documented and not well understood.

A key assumption in an ANP2P DHT protocol, such as Freenet, is that the routing table of a node is not known to its peers and cannot be manipulated by peers. However, we discovered that nodes in Freenet actually share their routing tables with their direct peers. A Freenet node knows its direct peers and the peers of its direct peers at two-hops away. Nodes share their routing tables to address local minimum issues and improve routing performance. An attacker can use the shared routing information to actively affect the routing table of a victim node with collaborating attack nodes. We developed a Routing Table Insertion attack that exploits this information and will discuss it more in Section 4.5.

The original Freenet design proposes a protocol that nodes use as part of the network joining process [19]. The goal is to use peer nodes in the network to calculate the location of the node joining the network. This location is used as the node's address. This was done to address the open research issue about secure assignment of node identities in Section 2.5.5. We identified several issues with this protocol based on the design specification. Colluding peers could manipulate the protocol to return an attacker-specified location. Another issue was there is no way to enforce that the joining node used the location calculated for it. Ultimately, the protocol to calculate the joining node location was not actually implemented in the protocol (or it was taken out in the version we studied). Currently a node can specify any location as its node address.

4.4 Traceback Attack

The first protocol design weakness that we identified was a Traceback attack. The Traceback attack allows an adversary to determine which nodes in a network have routed a particular message. This allows an adversary to find the subset of nodes that handled a particular message. Then using that subset, some additional topology information, and the routing algorithm used, an adversary can find the sender of the message. This attack was created as a joint effort, led by our colleagues Tian and Duan [80, 81, 82] at Florida State University. The Freenet development team has partially

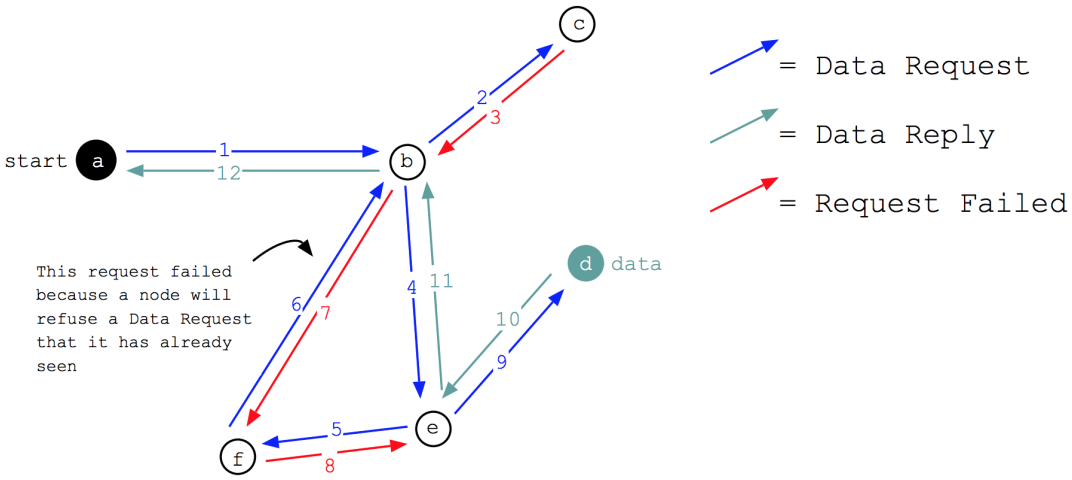

Figure 4.1: Freenet Routing. Clarke I, others. A distributed decentralized information storage and retrieval system. Undergraduate Thesis. 1999.

patched this vulnerability after we identified it.

Every message sent in Freenet has a unique identifier (UID) associated with it. When a message passes through a Freenet node, that node will store the message UID. This UID can then be used to prevent cycles in Freenet routing. Figure 4.1 depicts an example of Freenet routing. Node a is the message sender and node d is the target node to be routed to. The path that the message takes is indicated by the numerically increasing arrows. The interesting part of Figure 4.1 is data request 6. Node b has already stored the UID for the message being routed, so when node b receives 6, it replies with a FNPRejectLoop (7). The FNPRejectLoop message tells node f that the message has already been routed through node b. The idea of the attack is to leverage the information returned from message replies like 7 to determine which nodes a message has been through.

Figure 4.2 outlines the basic components needed to perform the Traceback attack. Node e represents an attacker's node that is on the network listening to the messages passed through it. When node e sees a message that triggers the attack, it will send the message M to the Attack Nodes (5). The Attack Nodes are a set of nodes that are under the control of the attacker. The Attack Nodes have a direct connection to all of the nodes in Freenet. After the Attack Nodes receive the message from node e, they will then try to determine the nodes that the message was routed through. The Attack Nodes will then broadcast the message M with a HTL count of 1 to all the nodes in the Freenet network (6) and wait for the responses. The Attack Nodes only care about FNPRejectLoop replies. These replies can be seen as 7 in Figure 4.2. The FNPRejectLoop replies tell the Attack Nodes that the message was routed through the set of nodes $\{b, a\}$. The sender of the message is then a member of the node set discovered by this attack.

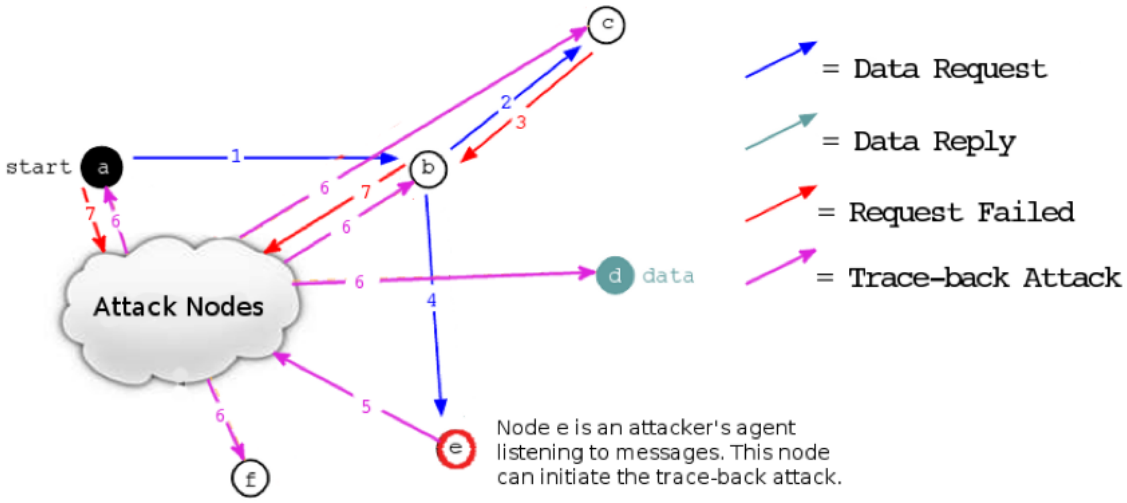

Figure 4.2: Traceback attack.

There is another issue that needs to be overcome for this attack to work. As previously mentioned Freenet uses a probabilistic decrement of 25% when the HTL reaches 1. This is an issue because we only want to check if a single node has seen a given UID, and we do not want to pollute other nodes with the UID. We leveraged a bug in the Freenet protocol to force a message to stop at the first node it was sent to and not be forwarded further. An invalid target address is passed in the message. The node will first check if it has seen the given UID in the message. If it has, it will return a FNPRejectLoop message. If the node has not routed the UID, then it will try to validate the target address and fail. This causes the node to drop the message. This behavior allows an attacker to determine if a specific node has already routed a message with a given UID or not.

The next issue that needs to be addressed for the Traceback attack to work is related to which nodes an attacker is directly peered with. The attacker will only be able to determine if a node has routed a message if they have a direct connection to that peer node. If the attacker is missing connections to some nodes, then they may not be able to determine the complete set of nodes that a message was routed through. This leads into the next attack we developed, the Routing Table Insertion attack, in Section 4.5. A comprehensive write up of the Traceback attack and its variations can be found in a technical report[4] and the publications [80, 81, 82].

In response to the Traceback attack finding, Freenet implemented a mitigation. Each Freenet node will now delete a message UID from its data store once the node has processed the message's reply. This means that it is still possible to perform the Traceback attack, but it can only be done within a time bound window. An attacker's node can slightly extend the length of this window by dropping or delaying messages that the node intercepts. The mitigation increases the difficulty of the attack, but it does not completely mitigate it.

4.5 Routing Table Insertion Attack

The Routing Table Insertion (RTI) attack exploits common performance improving mechanisms in order to attack a victim node's routing table. An adversary can use the RTI attack to insert malicious nodes into the routing table of a victim. This attack is used to enable other attacks like the Traceback attack described in the previous section. More information on the RTI attack can be found in the publication [2].

The first piece of information we need to perform an RTI attack is the network topology. There are two methods for collecting this information. 1) There is an undocumented debug function in Freenet called probe. This probe function takes a data location as an argument and will route a probe request based on the given data location. The probe function returns the node location of all the nodes it passes through. 2) Topology information can be collected from collaborating attack nodes. Each node knows $\approx p^2$ other node locations, where p is peer count. In Freenet, the maximum peer count is 40, so a node has the possibility of knowing 40^2 other peer locations. Collaborating attack nodes can merge their known routing information to build a network topology (or at least a sub-set of it). A complete network topology is not required to perform the RTI attack; however, it increases the effectiveness of the attack.

Assume we have learned the network topology. More information about our assumptions can be found in Chapter 3. Given a target node, we can use several attack nodes to collaboratively insert an attack node into the target node's routing table. This is done by exploiting Freenet's routing table management strategy. The routing table will periodically replace the least recently used peer with a new peer after the node has successfully handled 10 messages. The basic idea is as follows.

There are three basic components needed in the RTI attack: insertion node, query node, and intersection node. An insertion node is used to insert keys into the intersection node (along the insertion path). The query node is used to request the keys inserted in the intersection node (along the query path). Using these three components, an attacker can identify target (victim) nodes that are vulnerable to the RTI attack, and then insert an attack node into their routing tables. The attacker will control the insertion and query nodes, then use route prediction to find the intersection and target nodes.

In Figure 4.3, A_i is the insertion node, A_q is the query node, I is the intersection node, and T is the target node. (1) the attack node A_i can insert keys into node I via the insertion path without node T on the path. (2) Another attack node A_q can fetch the inserted keys with a query path from A_q to I with node T on the path. Note that the insertion path from A_i to node I is not overlapped with the query path from A_q to I, except at node I. The reason behind this requirement is that Freenet performs extensive caching to improve both performance and anonymity. When the insertion path and the query path are disjointed, no cached copies of the newly-inserted keys are on the query path. Note that only attackers know the IDs of these keys. As a result, caching does not affect this attack. To identify the intersection node, the insertion path, and the query path, we

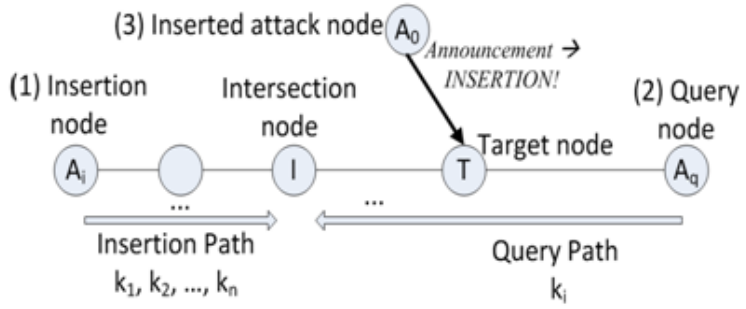

Figure 4.3: Illustration of Routing Table Insertion (RTI) attack.

have developed a route prediction model, presented later in this section.

To insert an attack node A_0 into the routing table of target node T, we choose multiple files (e.g., 30) with routing keys that are mapped to intersection node I and insert them via insertion node A_i. We found in practice that letting $A_0 = A_q$ is the easiest approach. After these insertions, we have newly inserted files that are located at node I. We then request these files from query node A_q. After we have successfully retrieved the files for a number of times more than the minimum threshold (the current default is 10), we let attack node A_0 announce itself to the network with an identifier that is close to node T. Based on the peer replacement policy, some existing peers of T may be dropped and attack node A_0 is likely to be accepted as a peer by T. If this fails the first time, we simply repeat this attack until A_0 is accepted by the target node as a peer.

We can maintain our attack node A_0 in T's routing table by frequently querying known files at A_0 from other attack nodes via node T. Due to the least recently used replacement policy, A_0 will not be replaced in T's routing table. In addition, once we have A_0 in T's routing table, it becomes much easier to insert other attack nodes into the table. We can then completely surround T with our attack nodes to perform an eclipse attack [77]. Consequently, we can completely control messages in and out of target node T, determine if a request is from T, or determine which file T currently holds, and break the anonymity promise.

Attack-Key Database. When inserting into an intersection node, the files must have a specific data location range. In order to expedite this process, we pre-compute the data location hashes of a large number of files, with a sufficiently large coverage of the complete identifier range $[0, 1]$. Note that, due to the nature of good hash functions (uniform distribution of hash values), we do not need to have sophisticated file structures and contents to cover the complete identifier range reasonably well. Actually, all files are one-line text strings, and we only need to slightly change the text string in different files to obtain completely different file identifiers. We have constructed an attack-key database with 30,000 pre-computed hashes, which easily covers the 50 nodes in our test-bed. For a large system, such as current Freenet with $\approx 3,000$ [31] nodes, a larger attack-key database would be needed.

Table 4.1: Reconstructed routing table of node H.

Node Location	Destination Peer	Direct Peer	Tie Count
0.1	B	D	1
0.15	K	K	1
0.25	A	D	1
0.35	L	K	1
0.4	I	D,K	2
0.45	E	E	1
0.5	D	D	1
0.6	C	E	1
0.9	J	E,K	2

Moreover, the current Freenet code allows a node to select its own node location. This means we can insert an attack node to any location on the network. This can be used to strategically place attack resources where the RTI attack will have the most effect.

4.5.1 Routing Prediction Model

To facilitate the RTI attack, we have developed a route prediction model based on the Freenet 0.7.5 routing algorithm and use it to figure out which nodes can be attacked.

Assumptions. The first assumption we make is that the topology (or its subset) is available. We can derive node routing tables from the topology, and we can then predict the routes of a request from a given source to its destination(s). Route prediction works on a topology subset with lower accuracy. The second assumption is that we will use a set of attack nodes running our revised code; all other nodes run standard Freenet code with the standard configuration settings.

Building Routing Tables. The route prediction model works by predicting all possible routing paths for a given network topology. We represent the topology as an undirected graph and use this graph to build node routing tables. A node's routing table includes all direct peers at one-hop away, and all indirect peers at two-hops away. Figure 4.4 has an example of a small topology. Using this topology, it is possible to construct node H's routing table. Table 4.1 is the routing table of node H. Details on how the routing table is constructed from a topology can be found in [4]. In Table 4.1, the Destination Peer column lists all of the peers that node H is connected to within two hops. The Direct Peer column is the direct peer that is routed to in order to get to the destination peer. A routing tie occurs when there are multiple peers in a direct peer entry. An example of a routing tie is the entry for destination node I.

A routing tie is resolved based on which peer has the longest life in a routing table. This is a local variable that we do not know, but we can try both paths in our attack because routing ties are not common on a path. Other routing details are omitted here. Based on the routing algorithm and a network topology, we can find a set of insertion nodes, intersection nodes, and query nodes

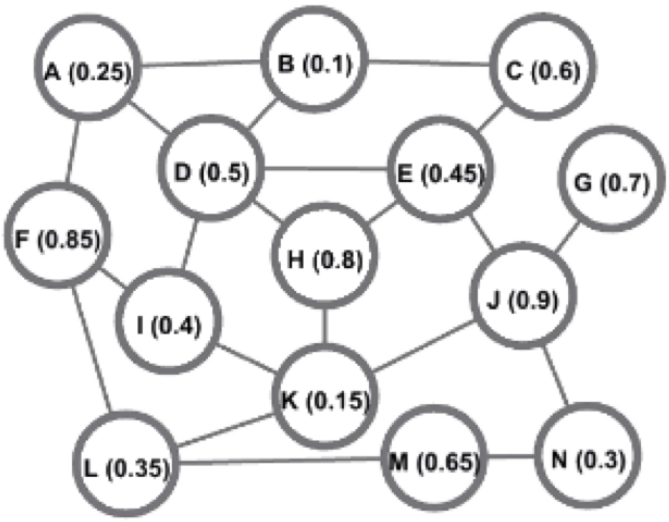

Figure 4.4: Example topology. Node location in parentheses.

to attack a target node.

Routing to the next peer. Once we have a node's routing table, we can predict the next peer to route to for all possible data locations coming into the node. A request is routed based on its data location. This location is compared to peer node locations to find the next hop. We do not need to know the data location in advance; instead, we predict the next hop for the entire data location range $[0, 1]$. This is similar to determining prefixes in an IP forwarding table. A node's routing table tells us all possible next hops, which allows us to take the data location as an input and map it to the next hop. We can then chain multiple nodes' routing tables together to get a full path prediction. Table 4.2 shows the complete routing table from node H for paths of length three (HTL=3) for the network topology shown in Figure 4.4.

We give the detailed prediction algorithm and examples in [4]. This algorithm recursively traverses all possible paths from a current node to a given length. However, there is a limitation to this routing prediction algorithm. It assumes that a node will route to its first choice node, and that is not always the case. Network issues, peer node responsiveness, and other factors that would cause a peer node to not respond will cause a node to try the next closest peer.

We have conducted extensive evaluation of the proposed scheme to validate the effectiveness of the RTI attack and route prediction model. We have conducted large tests to further analyze attack costs and develop countermeasures. We constructed a test-bed to run our Freenet experiments using a combination of VMware virtual machines and simulation. Details on our experiment environment can be found in [4].

Table 4.2: Predicted routes from origin node H with a length of three.

Data Location Range	Predicted Path
0.0–0.125	$\rightarrow H \rightarrow D \rightarrow B$
0.125–0.2	$\rightarrow H \rightarrow K \rightarrow J$
0.2–0.3	$\rightarrow H \rightarrow D \rightarrow A$
0.3–0.325	$\rightarrow H \rightarrow K \rightarrow J$
0.325–0.375	$\rightarrow H \rightarrow K \rightarrow L$
0.375–0.425	$\rightarrow H \rightarrow K(Tie = 2) \rightarrow I$
0.425–0.45	$\rightarrow H \rightarrow E \rightarrow D$
0.45–0.475	$\rightarrow H \rightarrow E \rightarrow D$
0.475–0.525	$\rightarrow H \rightarrow D \rightarrow E$
0.525–0.55	$\rightarrow H \rightarrow D \rightarrow B(Tie = 2)$
0.55–0.65	$\rightarrow H \rightarrow E \rightarrow C$
0.65–0.75	$\rightarrow H \rightarrow E \rightarrow J$
0.75–0.775	$\rightarrow H \rightarrow K(Tie = 2) \rightarrow J$
0.775–0.875	$\rightarrow H \rightarrow K(Tie = 2) \rightarrow L(Tie = 2)$
0.875–0.0	$\rightarrow H \rightarrow K(Tie = 2) \rightarrow J$

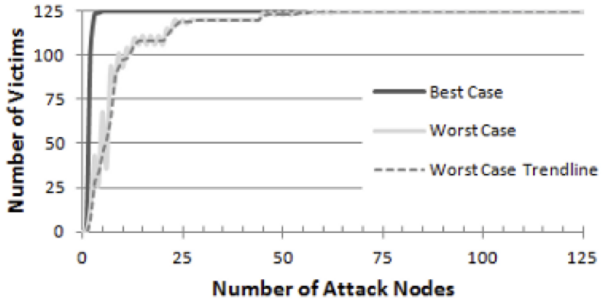

Figure 4.5: Effectiveness of RTI attacks on a 125-node network.

4.5.2 Effectiveness of the RTI Attack

We have carried out the RTI attack on our local testbed. Figure 4.5 shows the effectiveness of RTI attacks on a 125-node experimental network. The x-axis shows the number of attack nodes. The y-axis represents the number of victims out of the total 125 nodes. The two curves show the best case and the worst case for the numbers of victims for each set of attack nodes. With a set of ten attack nodes in the best cases, we can target almost all of the 125 nodes with the RTI attack (including the attack nodes); in the worst case, we still can cover about 80% of the nodes. As discussed before, once we have one attack node directly peered with a victim node, it is very easy to insert more attack nodes as its direct peers and conduct other attacks. Details on how the RTI experiment was performed can be found in the technical report [4].

Our analysis results give us effective ways to study an anonymous network. For example, given

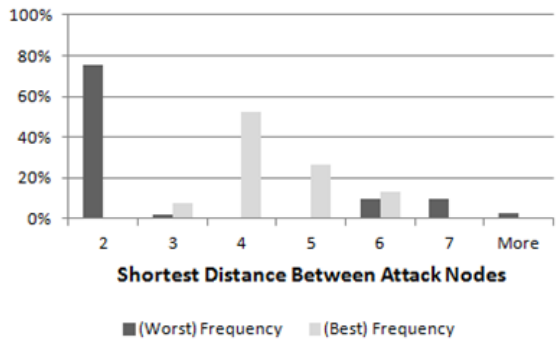

Figure 4.6: Percentage distribution of the shortest distance between attack nodes when the maximum RTI attack distance is seven. The best attack nodes and worst attack nodes are shown.

a set of attack nodes on a network, we can figure out which node we can attack quantitatively. Given a target node, we can figure out the time needed and the difficulty in compromising its routing table based on a set of attack nodes. We can also build different Traceback solutions to figure out which node inserts or queries a file. We present a Traceback attack [80] to detect the query sender with a high probability, even when it only requests a file once. Many other advanced attacks can be developed based on the RTI attack.

4.5.3 Attack Pair Analysis

The RTI attack requires at least two malicious nodes (insert node and query node). However, how these two nodes are selected has a big impact on the number of victim nodes that they can target. To identify the properties of a "good" attack pair, we constructed an experiment that generated 234 random Kleinberg topologies [71] with different node counts, peer counts, and HTLs. We then identified the best and worst attack pairs based on the number of their victims.

We found that the shortest distance between a pair of attack nodes has a significant impact on the number of victims they can reach. Figure 4.6 shows the distribution of the shortest distance between the best and worst attack pairs in the random topologies, where the maximum attack distance is seven (HTL=7). If a pair of attack nodes is more than seven hops away from each other, then they cannot reach each other. The x-axis is the shortest distance between a pair of attack nodes. The y-axis is the frequency of each case in terms of the percentage of the entire population.

Notice that in Figure 4.6 all of the distances between the best-case attack pairs are between the minimum attack boundary of two and the maximum attack boundary of seven. We found that keeping a distance between the minimum and the maximum generally improves the number of victim nodes that a pair of attack nodes can reach. Some of the distances of worst-case attack pairs are larger than the maximum HTL. These attack pairs are too far away from each other, and

they do not have any victim nodes for a given HTL. Most of the distances of worst-case attack pairs are the minimum of two, i.e., they are indirect peers. This means that the routable paths from both nodes are largely overlapped. One key feature of the RTI attack is to find an intersection node of two unique paths from the attack pair, i.e., both attack nodes can route messages to the intersection node. We can use the intersection node to manipulate the routing table of any nodes on the paths to the intersection from attack nodes. If both nodes have similar routing paths, this greatly reduces the number of intersections and the number of potential victims. We have also performed this experiment with a maximum attack distance of nine (HTL=9). The distributions of the shortest paths between attack nodes had very similar properties to Figure 4.6.

Guidelines for picking good attack pairs For a given topology with a given file access pattern, there is an optimal selection of attack nodes. However, it may not be feasible to use the optimal solution in practice. Therefore, we examine our simulation data to discover useful heuristics for attack node selection. In summary, the two nodes with the worst attack power are either too close (e.g., direct peers) or too far away from each other (the distance between them is near or over $2 * MaxHTL - 1$). Moreover, the minimum distance between the attack pairs should be at least three hops. These observations help us place attack nodes on a network to achieve the maximum damage on anonymity.

4.5.4 Examining the coverage of the RTI Attack

Another way to examine the potential damage of the RTI attack is to identify the set of victims for a given set of attack nodes. We have conducted a set of simulation tests and examined the actual attack cases. The results are shown in Figure 4.7. The x-axis is the percentage of attack nodes randomly selected on a network; the y-axis is the percentage of victim node on the network. As the percentage of attack nodes increases, the number of victims grows exponentially, i.e., an attacker can use a small percentage of all nodes to perform RTI attacks on the rest of the population.

Figure 4.8 shows the attack resources needed to achieve 100% network coverage when using the attack pairs with the minimum coverage. Notice that as the peer count and HTL increase, the necessary attack resources decrease. This means that fewer resources are needed to perform the same RTI attack when the attack nodes are better connected.

4.5.5 Evaluation of Route Prediction Model

We have evaluated the prediction model on our local testbed. Figure 4.9 shows the probability that we can find the correct routing path on a network. We first run these tests on our testbed to obtain the ground truth. We then use our prediction model to analyze the network and predict insertion paths. Here we use 75 to 200 nodes to build 5 random topologies, each with different configuration parameters. We then vary the number of peers per node and the maximum HTL to examine how effectively the route prediction works. Overall, we can find the complete insertion

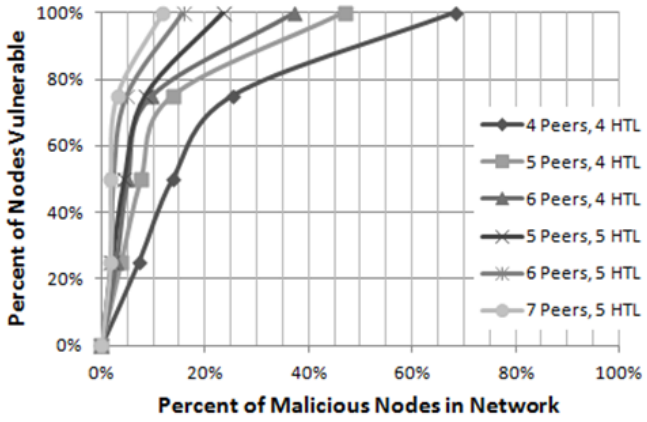

Figure 4.7: RTI Attack Coverage. (With 10% attack nodes, more than 50% nodes become victims on random topologies.)

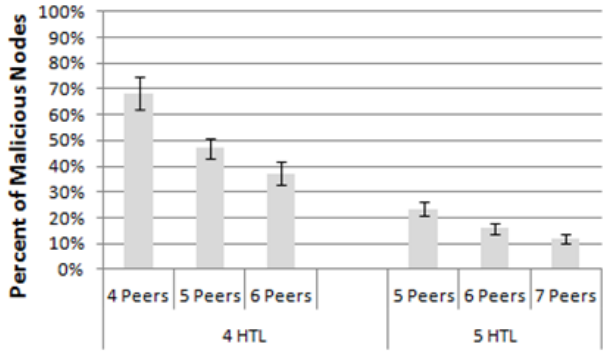

Figure 4.8: Average number of attack resources needed to target 100% of the nodes in a network using the attack pairs with the minimum coverage.

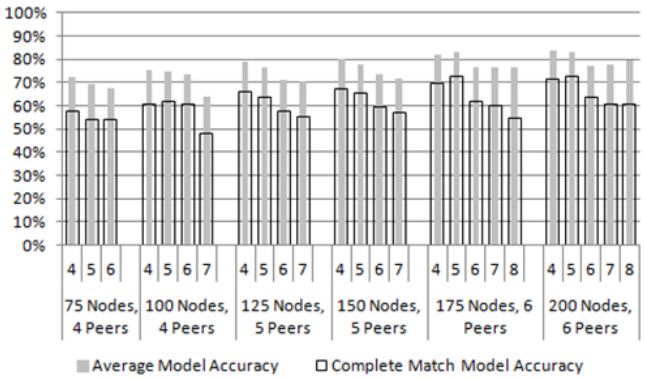

Figure 4.9: Accuracy of our route prediction model compared to actual routes by HTL.

path with an accuracy range of 48% to 72%. This means that we can find roughly more than half of the complete insertion routes correctly. In addition, prediction errors mostly occur close to the ends of predicted paths. That is, when we use the prefixes of these paths for attacks, we can still achieve higher success rates. Clearly, for short paths, the prediction model performs better. Figure 4.9 shows two different bars for each experiment configuration. The solid grey bar shows the average of how accurate our model is. For example, if our model matches the first 7 out of 10 nodes on an actual path, then our accuracy is 70% for that instance. The black outline is the percentage of paths where the prediction matched the actual path completely. As the number of nodes increases there is very little impact on the accuracy of the route prediction model. However, as we increase the HTL the accuracy decreases. This is because as we increase the HTL, it increases the number of predictions the route prediction model must make. Although these preliminary results certainly cannot prove the complete correctness of the prediction model, it gives us concrete evidence that the prediction model is valid for a majority of cases.

As each node runs the routing algorithm independently based on its local view of the network, our prediction algorithm cannot predict the exact route every time. However, we can predict each route with a deterministic probability depending on the topology of the network. On our 200-node testbed with a number of random topologies, we can predict most paths correctly with a reasonable probability. When we use these predicted insertion paths and query paths in the RTI attack, we only need a few tries to make it work.

4.5.6 Mitigations

We have proposed a random routing scheme [6] and a Look Ahead Hint mechanism [5] to mitigate the RTI attack.

Randomized Routing

To mitigate the RTI attack, we need to break one or more of its basic assumptions. After carefully analyzing these requirements, we found that some of them are basic P2P properties and have very little room to explore, because information sharing among peers is usually minimized for anonymity. Reducing an attacker's ability to predict routing paths is arguably the most feasible way to counter the RTI attack, as long as the routing performance is not seriously damaged. In the following, we will investigate the impacts of adding randomness to the greedy routing algorithm and evaluate both the performance and anonymity effects of this approach.

Randomized routing reduces an attacker's ability to predict routing paths. If the attacker cannot accurately predict the routing path, they will not be able to perform the RTI attack as illustrated in Figure 4.3. There are different ways to add randomness into a routing algorithm. For example, GNUnet's R^5N routing algorithm [33] adds randomness to the initial portion of a path by splitting the path of a message into two phases based on the message's HTL counter. In the first phase, a message is randomly routed for a fixed number of hops (relative to the estimated network diameter). In the second phase, greedy routing is used. Since an attacker can simply bypass the first phase of R^5N in the RTI attack, we have to consider a more generic case: adding randomness at each node independently, instead of based on HTL counters. In this scenario, each node randomly routes a message with a given probability that is not affected by the behavior of other nodes in the network, i.e., a message may be randomly routed at any node on its path. To illustrate the basic performance of this method, we assume that all nodes in the network have the same probability in our simulations.

It is extremely difficult to define a generic anonymity metric on P2P systems, so we propose to use the expected number of times that randomized routing will occur on a message forwarding path to quantify the anonymity strength of the randomized routing. We can estimate this number based on the length l of the forwarding path of a message and the probability p that a node may randomly route the message, as function:

$$R(p, l) = p * l \tag{4.1}$$

We use this simple estimation for comparison of the basic effects. A more elaborate R can be defined based on knowledge about the message path and the network churn. We also use a function Q to predict the effect of randomized routing on the RTI attack.

$$Q(p, l) = (1 - p)^l \tag{4.2}$$

The function Q will take the length l of the forwarding path of a message and the probability p that a node may randomly route the message, and it calculates the probability that randomized routing will not affect normal route predictions (predictions based on only greedy routing).

On Random Topologies. Figures 4.10 and 4.11 show the performance of randomized routing on random topologies. Each series represents a given randomized routing probability (RRP). The x-axis is the average network path length between nodes on a random topology. It is a graph property that is not affected by routing. We use it as the x-axis because it allows us to combine the node count and node degree variables into a single value. This reduces the dimensions of our output data and shows the combined effect. A large average network path length means that a topology is more spread out, with a large diameter. The y-axis is the average routing path length (i.e. search path length).

We will first discuss the performance impacts of the look ahead. Figure 4.10 shows the performance of randomized routing when one-hop look ahead is used, and Figure 4.11 shows two-hop look ahead. By default, Freenet is configured with two-hop look ahead; however, it is possible for nodes to be configured with only one-hop look ahead. As expected, two-hop look ahead greatly increases the routing performance. This can be observed by comparing the base case for routing (0% RRP) in Figure 4.10 and 4.11 (note the y-axes are not on the same scale). In addition, using two-hop look ahead reduces the performance degradation as the randomized routing probability is increased. The differences between the average routing path lengths of the various randomized routing probabilities and that of the base case are much smaller.

In Figure 4.10, the series are not smooth as we move across the x-axis. This is also the case in Figure 4.11; however, it is much less pronounced. The spikes are a result of using the average network path length as the x-axis. This metric combines the node count and node degree variables together, and as a result the node count does not grow linearly as we move across the x-axis. The random topologies in Figure 4.10 and 4.11 are sensitive to this variation in the growth of the node count. The spikes in the series are places where there is a higher node count than the immediately adjacent data points. We found that as we increase the look ahead value on random topologies, it decreases the sensitivity to the variation in node count. Also, the small-world topologies in Figure 4.12 and 4.13 are not sensitive to the node count variation.

As the average network path length increases in Figure 4.11, the average routing path length also increases as expected. Increases in the average network path length mean larger network diameters, which means that on average message routes are longer. This behavior is common across all simulations and can be seen in all performance-related figures shown here.

As shown in Figure 4.11, as a node's randomized routing probability increases, the average routing path length also increases. When we look at the topologies that have an average network path length of 3.7 hops in Figure 4.11, the average routing path length is 7.6 hops with 0% RRP. However, with the 20% RRP, the average routing path length almost doubles to 13.51 hops. Consider that the 0% RRP is the base line. We calculate the expected times that randomized routing may occur using function R from the previous section, $R(.2, 13.51) = 2.702$ This means that with the 20% RRP, the path grows an additional 5.91 hops (> 2.7 in the above). This suggests that

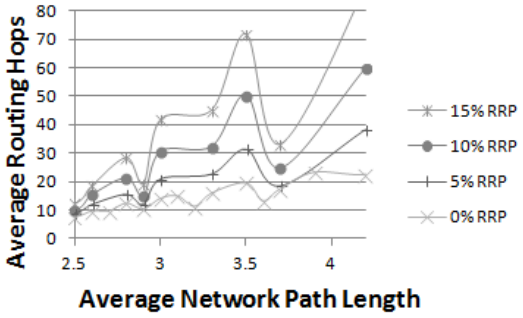

Figure 4.10: Routing performance in random topologies with one-hop look ahead.

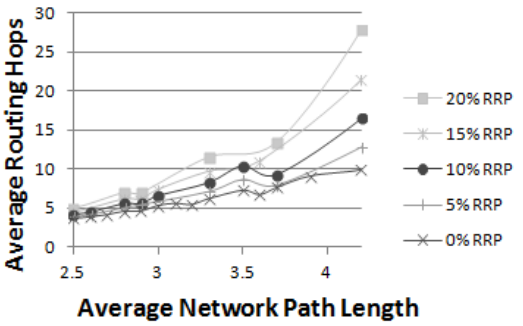

Figure 4.11: Routing performance in random topologies with two-hop look ahead.

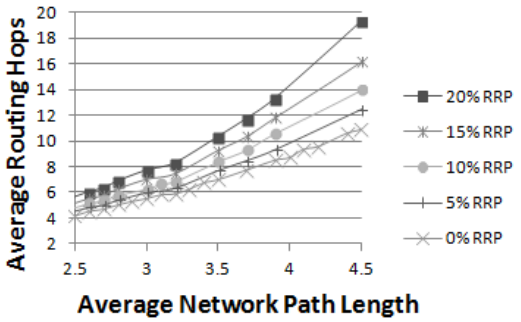

Figure 4.12: Routing performance in small-world topologies with one-hop look ahead.

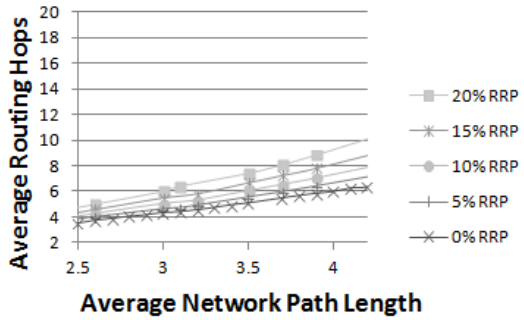

Figure 4.13: Routing performance in small-world topologies with two-hop look ahead.

every time random routing occurs, it causes a message's forwarding to regress, and adds several hops to the routing path.

Freenet currently uses a maximum HTL value of 18. In Figure 4.11, most of the average routing path lengths are under the 18-hop ceiling (except for the cases of 15% and 20% RRP on topologies with an average network path length over 4). However, we need more than just the average path length to be less than 18 hops. We also want the range of path lengths within two standard deviation to be under the 18-hop ceiling. Two standard deviations account for 95.45% of the average path lengths of randomized routing. When we include two standard deviations from the average path length of randomized routing, we find that the 10% randomized routing is above the 18-hop ceiling.

On Small-world Topologies. Figures 4.12 and 4.13 show the average routing performance of randomized routing on small-world topologies. We notice that the trends are similar as on random topologies as shown in Figure 4.11. As the average network path length increases, so does the average routing path length. Adding more randomness to the routing algorithm also increases the average routing path length. However, we see much shorter average routing path lengths on small-world topologies compared to random topologies.

When we compare the baseline case (0% RRP) between random and small-world topologies

(Figure 4.10 compared to Figure 4.12, and Figure 4.11 compared to Figure 4.13), we see better performance on small-world topologies. The number of extra hops added to paths due to randomized routing is decreased, compared with the average routing path lengths on random topologies. When using two-hop look ahead on small-world topologies, the average path lengths in all cases (including their standard deviations) fall under the 18-hop ceiling.

Anonymity. We estimate the number of times randomized routing is triggered using function R, the randomized routing probability, and the average routing path lengths. Higher return values from function R indicate more random routing in a path and, in turn, a less deterministic route. The concern is that there is no guarantee random routing will occur, or where it will occur along the path.

In our work on the RTI attack, we were able to correctly predict routing paths with a simple route prediction model that had an accuracy that ranged between 48% to 72% depending on network variables. Let us assume for simplicity that an adversary was able to 100% correctly predict routing paths based on the greedy routing algorithm. The effect that randomized routing would have on those predictions can be expressed with the function Q that is defined in the previous section. For example, given a 20% RRP and an average routing path length of 8 hops (taken from Figure 4.13), we have $Q(.2, 8) \approx .17$. In this case, there is roughly a 17% probability that the route predictions will still be correct.

We found that small-world topologies can handle randomized routing much better than random topologies. A 20% randomized routing on a small-world topology with two-hop look ahead will still achieve an average routing path length within 18 hops (including its standard deviation). For this case, we expect that random routing occurs once every five hops on average. On random topologies with two-hop look ahead, the only case that can achieve an average routing path length completely under the 18-hop ceiling including standard deviation was the 5% randomized routing. However, in this case, we expect random routing to occur only once every 20 hops on average.

Randomized routing may help us break deterministic routing paths, but it comes with a heavy routing cost. If not carefully implemented, adding small amounts of randomness may greatly decrease the routing performance. However, our simulation results show that applying randomized routing on small-world topologies will not result in heavy performance degradation. We have examined one type of random routing in this paper. Future work includes further investigation of more advanced randomized schemes to explore basic ANP2P properties in order to maintain anonymity without heavy performance costs.

Look Ahead Hint

While the look ahead method improves routing performance, it also shares more identifiable node information with a large number of peers, which can be utilized in routing-based attacks. To address this issue, we developed the Look Ahead Hint (LAH) method. Instead of passing a node's

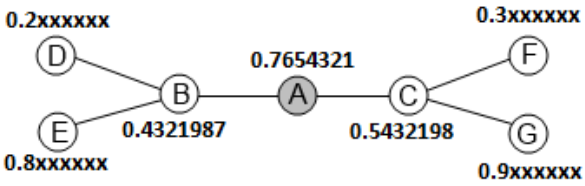

Figure 4.14: Example of look ahead hint (LAH) of size one.

full address like the original look ahead method, we only pass the most significant units of the address when the look ahead count is two or more. In other words, a node still shares its full address with direct peers one hop away, but nodes farther away only get the high-order portion of the node address. The low-order portion of an address is masked to hide its real values. We call a masked address a *hint*. A hint of size three means that all but the three most significant units of an address are masked. LAH relies on the direct peers of a node to mask its address before passing the address to other nodes at two or more hops away.

Let node n have address a. Let set P_1 denote its peers at one hop away, and let set P_2 denote its peers at two hops away. In LAH, n first sends the address a to the nodes in P_1. Each node n_1 in P_1 then only sends the hint of address a to nodes in P_2, the nodes at two hops away from n. Every node in P_1 gets n's full address a, and nodes farther away (in P_2) only get the hint for routing.

We made several assumptions for LAH. (1) The node addresses must be comparable. Given two addresses, we can not only test for equality but also compare which one is larger. (2) An address is still valid after we mask its least significant units. (3) Node addresses are evenly distributed, and so the most significant units of node addresses are also evenly distributed.

The size of significant units in an address depends on the system design, e.g., a system that uses a 16-bit unsigned short for an address may use a bit as a significant unit. Freenet uses floating-point numbers in the range [0, 1) to represent addresses. Figure 4.14 shows how LAH works on Freenet. In this example, we use a decimal digit as a significant unit. The most significant digit is the tenths place, and we use two-hop look ahead with a hint size of one (only using one significant digit). We assume that nodes have already propagated their hint information. Node A's routing table will have the full addresses for nodes B and C, and the hints of nodes D, E, F, and G.

We expect that LAH will not provide as much performance improvement as the original look ahead with full addresses, because we give nodes less routing information to work with. We use the average routing path length as a metric to measure the performance impacts of LAH. Intuitively, LAH provides more anonymity than the original look ahead method.

In order to quantify the anonymity provided by LAH, we calculate the probability that an adversary can identify a node in the topology using hints. Let u denote the number of possible values that a single significant unit can have, let b denote the number of significant units (i.e., hint size), and let t denote the total number of nodes in the network. We define function $F(u, b, t)$ to

estimate the chance that an adversary can identify a node in a network based on hints.

$$F(u,b,t) = \begin{cases} \frac{u^b}{t} & \text{if } u^b < t. \\ 1 & \text{if } u^b \geq t. \end{cases} \tag{4.3}$$

When $u^b \geq t$, the adversary has a very good chance of using the hints to identify source nodes, because hints do not help to hide node addresses in this case. We define $F(u,b,t) == 1$ to indicate a higher risk.

Now we can estimate the number of peers that a given malicious node can possibly have in its routing table. We then use the estimated node reach to calculate the amount of resources that an adversary needs to cover all edges in a topology. Let e denote the number of edges in a topology, let r denote the average peer reach of a malicious node, and let t denote the total number of nodes in the topology (including malicious nodes). We can estimate r using $\approx (p-1)^h$ where p denotes the average peer count (or degree). The value for r can also be measured.

We define function $A(e,r,t)$ to represent the percentage of the topology that the adversary must own in order to cover all edges.

$$A(e,r,t) = \frac{\frac{e}{(r-1)}}{t} \tag{4.4}$$

where $(r-1)$ represents the number of edges that a single malicious node covers, and $\frac{e}{(r-1)}$ is roughly the total number of malicious nodes required to cover all e edges. We then normalize it by dividing the total number of nodes t. This function is the lower bound of resources that the adversary must own, and represents the best case scenario, in which every malicious node is able to cover a *unique* set of edges, i.e., each edge is covered only once by a malicious node. For a malicious node to *cover* an edge, it must know the nodes at each end of the edge. The value of e can be measured on a given topology or estimated. When p is the average peer count, then $\frac{p \times t}{2}$ is roughly the number of edges.

This estimate will only apply to peers that are two or more hops away in a node's routing table. Now we can determine an adversary's ability to use the hints to identify nodes. We combine this metric with the anonymity metric $A()$. We define metric $H()$ such that: if $F(u,b,t) < 1$, we use the average node reach with one-hop look ahead to calculate the resources that an attacker needs, denoted as r_1. This is because LAH does not give an attacker enough information to identify the nodes that are two or more hops away; they are only left with the nodes at one hop away. If $F(u,b,t) == 1$, the attacker has a high probability to identify the nodes provided by the hints at two or more hops away. In this case, the anonymity provided by LAH will be the same as the original look ahead.

$$H(e,r,t) = \begin{cases} A(e,r_1,t) & \text{if } F(u,b,t) < 1. \\ A(e,r,t) & \text{if } F(u,b,t) \geq 1. \end{cases} \tag{4.5}$$

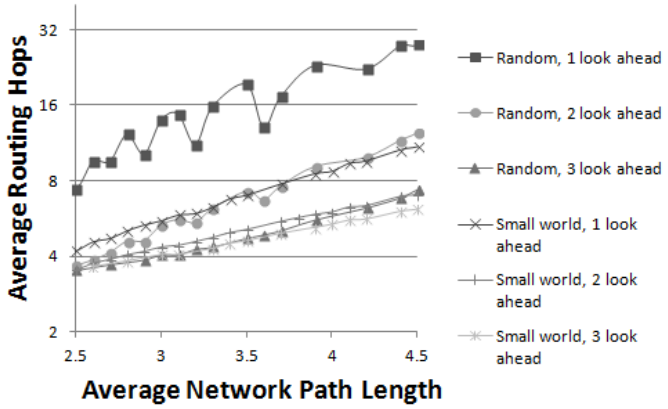

Figure 4.15: Look ahead routing performance on small-world and random topologies.

Let us explain this further with examples. Given two-hop look ahead and a hint size of one ($b = 1$) as shown in Figure 4.14, and given 100 total nodes ($t = 100$) and a significant unit of a decimal digit ($u = 10$), we have $F(10, 1, 100) = 0.1$. This means that an attacker has a maximum 10% chance to identify the actual node represented with a given hint. If we change the hint size to two ($b = 2$), then $F(10, 2, 100) = 1$, which means that an attacker has a 100% probability to correctly identify the actual node from hints, because hints does not help us hide nodes in this case.

However, the hint size also affects routing performance. If it is too small, it may cause a large number of routing ties; if it is too large, all anonymity benefits are lost. To address this issue, we define function $B(f, u, t)$ to find the optimal hint size.

$$B(f, u, t) = \lfloor (\frac{\log(t \times f)}{\log u}) \rfloor \tag{4.6}$$

Let f be the maximum desired probability that an attacker can uniquely identify a node from a hint. Let u be the number of possible values that a single hint can have, and t be the total number of nodes in a given network. $B()$ gives us the optimal hint size such that an attacker's ability to identify a node based on a hint is no larger than f. For example, given a network of 10,000 nodes and a decimal digit as a hint significant unit, an attacker's ability to identify nodes from a hint is $\leq 25\%$, meaning that an adversary has a one in four chance of identifying a node using a hint. Then $B(0.25, 10, 10000) = 3$, which means that a hint size of three is optimal based on the given conditions.

Experiment Analysis: Original Look Ahead Method. We first evaluate the performance of the original look ahead method, and then analyze its anonymity strength. We further discuss the trade-off between its performance and anonymity.

Performance Evaluation. Figure 4.15 shows that the routing performance on a random topology with two-hop look ahead (*Random, 2 look ahead*) is roughly the same as that on a small-world

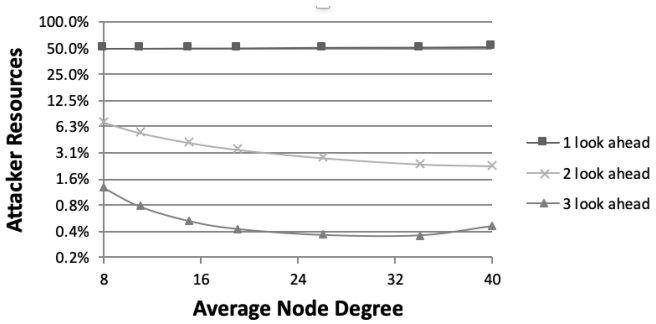

Figure 4.16: Estimated adversary resources needed to construct complete real-time topology.

topology with one-hop look ahead (*Small world, 1 look ahead*). This tells us that the small-world topology does in fact improve routing performance over a uniformly random topology. In addition, when additional look ahead hops are used on both small-world and random topologies, the average path length is exponentially decreased. In the paper [49], the authors obtained a performance improvement of 91% by going from one-hop look ahead to two-hop look ahead in a well-trained network, with a previous version of Freenet routing algorithm. We tested with the current improved routing algorithm and did not observe such a huge performance improvement in our simulations when increasing look ahead hops. Because the results are obtained under different routing algorithms, it is difficult to directly compare them. Furthermore, as the average network path length increases, so does the average routing path length. This property is true in all simulations. The performance of one-hop look ahead on random topologies in Figure 4.15 is not as smooth as the performance on the small-world topologies. However, a clear trend can still be observed. When we increase the look ahead value on the random topology series, it shows a smoother routing performance.

As the average routing path length increases in Figure 4.15, so does the standard deviation, e.g., the series of *Random, 1 look ahead* has the longest average routing path lengths and the largest standard deviations.

Anonymity Evaluation.

Figure 4.16 shows the estimated amount of resources that an adversary would require to construct a complete real-time topology. The series are plotted using metric $A(e, r, t)$. The values of e, r, and t are measured from the topologies used in the simulations. The y-axis is the percent of the total number of nodes that the adversary has to control in a network. Note that the y-axis is in logarithmic scale. We found that an adversary needs to control roughly 50% of a network in order to cover all edges when it has only one-hop look ahead information. This is a fairly challenging requirement on a large P2P network. However, when two-hop look ahead is used, the resources needed by an adversary drastically decrease to roughly 2% to 6% of the network. We tested two-hop look ahead with various topologies ranging from 3,000 to 10,000 nodes, with a mean node count of

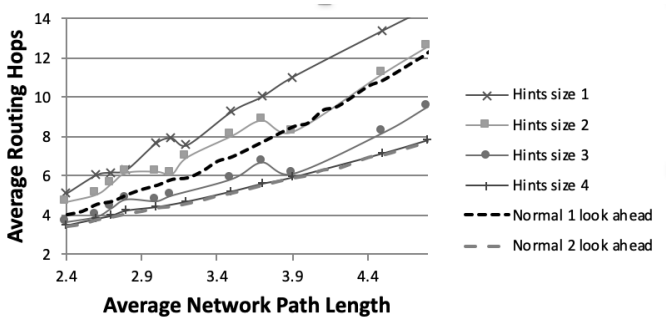

Figure 4.17: LAH routing performance on small-world topologies with two-hop look ahead.

5,400. In this best-case analysis, the attacker would need to control 108 to 324 nodes out of 5,400. The metric $A(e, r, t)$ is a simplified estimate, and it assumes that every attack node is capable of covering a *unique* set of edges in a network. This is not a realistic assumption. When attack nodes are added in a real world network, it is hard for an attacker to avoid duplicate coverage on a network without knowing its topology a priori.

The current look ahead mechanism on Freenet provides performance improvements, at the cost of sharing node address information with a potentially large number of nodes. Adversaries can use this information to learn more about the ANP2P network and mount attacks, such as the RTI attack. When two-hop look ahead is used, it greatly reduces the required resources for an attacker to harvest topology information. Based on the previous data obtained on real-world Freenet nodes, we can use a single node to harvest 596 unique node addresses on average, about 15% of the average size of 4000 nodes simultaneously online at that time (based on Freenet statistics). Figure 4.16 shows the tradeoff between performance improvements using look ahead and an adversary's ability to construct a complete real-time topology.

Experiment Analysis: Look Ahead Hint (LAH). We proposed LAH to balance performance and anonymity. We designed several experiments to test the performance and anonymity of LAH and compared it with the original look ahead method. In these simulations, we used a decimal digit as a significant unit, and varied the hint size as 1, 2, 3, and 4 digits.

Performance Evaluation. Figure 4.17 shows the average routing path length under LAH with various hint sizes on small-world topologies. All *Hints size X* series represent the routing performance when using two-hop look ahead with the hints of X digits. The *Normal 1 look ahead* series shows the performance of using the original one-hop look ahead, and *Normal 2 look ahead* series represents the performance of the original two-hop look ahead. We found that, as the hint size increases, the performance is closer to the *Normal 2 look ahead* series. This is expected since hints provide nodes with less information to route with. The best performance of LAH is close to that of its original look ahead counterpart.

The interesting part of Figure 4.17 is that when the hint sizes of one and two are used, they

Table 4.3: Maximum probability of uniquely identifying a node from Look Ahead Hint in our experiments in Figure 4.17.

X Hint size	Adversary's chance of identifying a node
1	$F(10, 1, 5400) = 0.185\%$
2	$F(10, 2, 5400) = 1.85\%$
3	$F(10, 3, 5400) = 18.5\%$
4	$F(10, 4, 5400) = 100\%$

perform worse than the original one-hop look ahead. This is because LAH may provide nodes with too little information to route with, and that can be more detrimental to routing performance than if they were not used at all. Smaller hint sizes cause more routing ties because they can only represent a small subset of the possible address values. In the current algorithm routing ties are randomly broken, which is why we saw worse performance than one-hop look ahead. Once we change the tie breaking policy in the simulator algorithm (by using one-hop node address information to break the ties that occurred at two or more hops away), LAH's performance is the same as one-hop look ahead.

In Figure 4.17, the hints of size three perform better than one-hop look ahead, but not as well as two-hop look ahead. When we use the hints of size four, the routing performance is equal to that of the original two-hop look ahead. We also performed these experiments on random topologies with three-hop look ahead. The results are similar to Figure 4.17 in all of our experiment variations.

Anonymity Evaluation. Table 4.3 shows an adversary's maximum chance to uniquely identify a node address from hints used in Figure 4.17. The average node count in the experiments is 5400 nodes. When metric $F(u, g, t) < 1$, it is difficult for an adversary to uniquely identify a node from just a hint. In these experiments we find that LAH does provide more protection to node addresses until we get to a hint size of four. With a hint size of four, we can uniquely identify a node because the hint size provides more unique values (10,000) than the total number of nodes in the system. When the hint size is four it no longer protects a node's address from being identified by peers two hops away. The 100% chance only means that node information may be exposed when using hints, but the attacker still needs to do extra work to identify a node. If we calculate the optimal hint size for our experiments, assuming the adversary will only be given a 25% chance of identifying nodes, it is $B(0.25, 10, 5400) = 3$. Table 4.3 verifies the result returned by the metric $B(f, u, t)$.

We can use the metric $H(e, r, t)$ to quantify the anonymity of LAH. Figure 4.16 represents the output of metric $H(e, r, t)$, and shows the resources that an adversary would need to construct a complete real-time topology. The hints of size one, two, and three will all have a resource requirement equal to the *one-hop look ahead* series. Note that even though we can achieve the anonymity equal to one-hop look ahead, we will still incur the overhead of two-hop look ahead in this example. As the hints of size four have metric $F(u, g, t) = 1$, they will have the same anonymity of original two-hop look ahead. In this case, it would be the *two-hop look ahead* series in Figure

4.16.

If a hint size is chosen carefully, it can provide better anonymity over the original look ahead technique. However, if the hint size is too large, any anonymity improvement provided by LAH is lost, and it provides the similar level of anonymity of the original look ahead. We found that the closer we can get the metric $F(u, b, t)$ to 1 without reaching it, the better performance we will achieve. In addition, if the size of the significant units used as hints can be reduced, it gives us finer-grained control over the trade-off between performance versus anonymity gained from LAH. Although it will not achieve the same routing performance as the original look ahead counterpart, it gets very close.

4.6 GNUnet Bloom Filter

The next protocol design weakness we identified is an attack on Bloom filters in the GNUnet protocol. GNUnet uses a Bloom filter to detect routing loops. The Bloom filter is included as part of every message that is sent. All *get* and *put* operations include this Bloom filter as part of the message passed between nodes. When a node receives a message, it will add itself to the Bloom filter in the message before forwarding it on. The danger of this behavior is that it provides an adversary with a potential set of nodes that have routed a given message. If an adversary receives a message, they can brute force the message's Bloom filter against known nodes. This assumes that the adversary has some means to harvest node identifiers. The Bloom filter can produce false positives, but it will never produce false negatives. This means that it will never exclude a node that has actually routed the message but could potentially include nodes that have not. We have only performed an initial analysis of this attack. The results of the initial investigation will be discussed first. Then the future work required to complete this analysis will be enumerated.

We used an equation, fpr, to find the false positive rate of the Bloom filter. The false positive rate will indicate how useful the Bloom filter object could be at identifying previously routed nodes. The lower the false positive rate is the more likely an adversary could use the Bloom filter to uniquely identify previous routed nodes. Given the size m of the Bloom filter, the k number of hash functions used, and the n number of elements inserted in the Bloom filter, we can find the probability of getting a false positive out of the Bloom filter. In this case, n is the number of previously routed nodes.

$$fpr = (1 - (1 - \frac{1}{m})^{kn})^k \tag{4.7}$$

There are better functions for determining the false positive rate of a Bloom filter than the fpr function we are using; however, we find that the simplicity of fpr is sufficient for our initial analysis. Based on the GNUnet implementation, we assigned $m = 128$, $k = 16$, and we varied n from zero to twenty-five. Figure 4.18 shows the results of our analysis. The figure shows that if the number of nodes that have routed a message is ten or less than the false positive rate is near zero. After ten

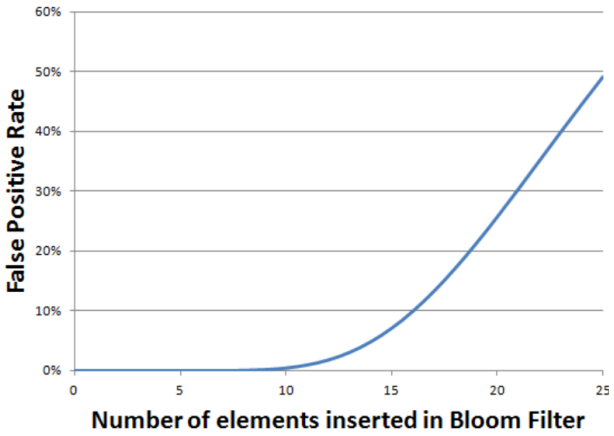

Figure 4.18: False positive probability of a GNUnet message Bloom filter.

nodes in the Bloom filter, then the false positive rate starts to climb. However, the path length of messages is limited by HTL in GNUnet, so the number of nodes in the Bloom filter should be at most ≈ 20.

Our initial research suggests that the Bloom filters can potentially be used by adversaries to reduce a message's sender set. However, addition work is required to determine the feasibility and effectiveness of this anonymity attack. First an in-depth analysis of the GNUnet protocol is required. GNUnet may have mitigating controls in place that reduce the risk of this Bloom filter attack. Next, a study of the feasibility and resources required for an adversary to collect all node public identifiers is required. As part of this study, the average lifetime of a node's public identifier should also be considered. Finally, the accuracy of the GNUnet Bloom filter attack will need to be evaluated. Accuracy refers to the attack's ability to correctly identify the nodes that previously routed a given message from other nodes.

4.7 Summary

In this chapter, we provided a brief overview of how Freenet works. We performed an in-depth analysis of the protocol specification for Freenet and a preliminary analysis for GNUnet. We derived the design and implementation specifications from available literature, review of source code, and instrumenting the executables. Then we studied the security and anonymity implications of the protocol specification. We found several interesting findings: the Traceback attack, the Routing Table Insertion attack, and information leakage with the GNUnet Bloom Filter. Based on the findings, we developed a novel attack against Freenet anonymity using a combination of the Traceback attack and the Routing Table Insertion attack. We also performed an initial investigation of a potential vulnerability in GNUnet's design. This resulted in the Bloom filter attack. Further

research is required to complete the GNUnet Bloom filter attack analysis.

The next steps in our research build on the work done in this chapter. We define ANP2P DHT design properties in Chapter 5. We then design and run an empirical methodology for evaluating the performance and design decisions in Chapter 6.

CHAPTER 5
DESIGN CHOICES

In this chapter, we will discuss the ANP2P DHT design choices we have observed. First, we will define the scope used during the study of the designs. Then each design choice will be defined along with its potential implementations. Table 6.1 has a high-level summary of the design choices.

In this research, we focused on the mechanisms and designs to route a message from a given starting node to a given target node. We have not considered design choices related to data storage, data retrieval, or data replication. All of these are important; however, we focused on the core routing capabilities of ANP2P DHT protocols. Almost all other protocol features are built on top of the routing feature.

Topology Type:

- Structured: The topology will maintain some invariant about how each node is connected.
- Semi-structured: The topology uses a best effort approach to maintain some invariant; however, it is not guaranteed that every node will comply.
- Random: Every node connection is chosen at random, and there is no predictable structure in the topology.

Routing Type: Routing type determines how a node will pick the next peer to route a message to from the available peer connections. The available peer connections are defined based on the topology type. Routing types are split into five categories: deterministic, greedy, semi-random, random, hybrid.

- Deterministic: Each node knows the exact path that a message must take to reach its target destination. A node will use its local peer information combined with the invariance of a structured topology to build deterministic routing paths. Deterministic routing paths have the benefit of routing performance guarantees.
- Greedy: Each node will try to make the best routing choice based only on the local peer information available. Greedy routing algorithms will provide best effort routing performance, but they cannot guarantee it. Greedy routing works relatively well with semi-structured topologies.
- Semi-random: A variable amount of randomness is introduced into a deterministic or greedy routing algorithm. This will make the routing path less predictable; however, it often negatively impacts the performance of a protocol.

Table 5.1: Design Choices used in empirical analysis

Design Choice	**Implementations**
Topology Type	Structured Semi-structured Random
Routing Type	Deterministic Greedy Semi-Random Random Hybrid
Hop-to-live	Deterministic Probabilistic Hybrid None
Routing Table Maintenance	Deterministic Random None
Backtracking	true false
Loop Detection	true false
Look Ahead	1 2 ... k
Size	n
Degree	d

- Random: Each node will choose the next peer to route to randomly. The amount of randomness used can range from none to completely random. Introducing randomness in the routing algorithm can have significant impacts on routing performance.

- Hybrid: Multiple other routing types can be combined. A common example of this is two phase routing. A message will be randomly routed for a given number of hops. Then greedy or structured routing will be used for the remainder of the message's routing.

There are also several routing mechanisms based on quality of service (QoS) that we will briefly mention. Some routing algorithms use QoS information such as observed bandwidth, success rate, and latency to break routing ties. The goal of these QoS routing decisions is to maximize QoS parameters (such as throughput, delay, loss, or jitter). In our experiments and models, we have considered both paths when a routing tie is present.

Hop-to-live: The hop-to-live (HTL) value is used to limit the resources a message will take when it is being routed. Without a HTL value a message could potentially be forwarded indefinitely under certain circumstances or it could be routed to all n nodes in a topology before it reaches its destination.

- Deterministic: Each time a message is forwarded to a new node, the HTL value will be incremented/decremented (depending on if the HTL counts up or down). The biggest risk with using deterministic HTL values is that it will tell a malicious adversary exactly how many hops away the original sender is. An adversary can also use deterministic HTL values to positively identify the original sender if the adversary is one hop from the sender when observing a message.

- Probabilistic: Each time a message is forwarded, there is a probabilistic chance that the HTL value will be incremented/decremented. This prevents an adversary from knowing exactly how far away the original sender of a message is. Even if the original sender is only one hop away from the adversary. An adversary can still build a probabilistic model of the distance between the adversary and the original sender.

- Hybrid: A combination of deterministic and probabilistic approaches is used. Probabilistic HTL is used at the beginning of message routing, and when the message HTL approaches the maximum value. Deterministic routing is used for all of the hops between the minimum HTL value and the maximum HTL value. Freenet uses an example of this.

- None: Structured topologies and routing algorithms have deterministic message delivery paths that will always be less than some constant number of hops. As a result, a HTL value is not needed. It is not possible for a message to be routed indefinitely.

Routing Table Maintenance: Each node will maintain a local routing table of peers that it can route to. Some ANP2P protocols will periodically remove a peer from a node's routing table and add a new peer. This is done to induce churn in the topology and to help prevent attacks that analyze long-lived nodes and circuit paths. The statistical intersection attack [23] is an example of one of these attacks. When the protocol triggers a routing table replacement, there are a few methods that can be used to pick a node to be replaced.

- Deterministic: Pick the peer to drop based on some property. For example, Freenet will replace the least recently used peer. The risk of using deterministic peer replacement is that it can be abused to force an attacker into a victim's routing table. See Section 4.5 for more details on our Routing Table Insertion attack that leverages Freenet's deterministic peer replacement policy.
- Probabilistic: Pick a peer at random to replace.
- None: Do not periodically replace peers in a node's routing table.

Backtracking: When a message is being routed, it is possible that the routing algorithm can hit a local minimum or a relatively isolated subgraph in the topology, resulting in the message hitting a dead end. The routing algorithm can either give up and report that the message failed to be delivered, or it can try backtracking and taking a different path. Protocols that use deterministic routing do not need backtracking enabled, but it is typically used in semi-structured networks to improve routing resiliency.

Loop Detection: When a message is being routed it is possible for it to get stuck in an infinite loop. The HTL value can be used to prevent resource exhaustion when this happens. In addition, loop detection can be used to prevent routing loops. We have observed two methods of implementing a loop detection mechanism. The first is for each node to maintain a list of previously routed messages and detecting if a loop occurred once a message reaches a node again. The second is to store the list of previously visited nodes as part of the message. The benefit of this is that a node can check if a message has already been routed to a given peer before it tries routing to that peer. Protocols that use deterministic routing do not need loop detection.

Look Ahead: Each node will only use local peer information when routing in ANP2P DHT protocols. If nodes do not have enough topology information, then they may not be able to successfully deliver a message, or the routing path can end up being longer than it needs to be. Look ahead can be used to expand a node's peer information. Instead of just looking at the node addresses of direct peers, a node can use the addresses of nodes multiple hops away to make routing decisions. A look ahead of 1 would mean that a node can see all the peers that are one hop away (direct peers). Look ahead of 2 would be all peers within two hops of the node. The look ahead value can be extended to an arbitrary k number of hops.

Size and Degree: We found from our experiments in Section 4.5 that the size of a network (i.e., the total number of nodes in the network) and average node degree have a significant impact on overall routing performance. The network size n must be large enough to provide user anonymity in the crowd. At the same time, the node degree d (the number of peers per node) needs to be large enough to provide decent performance. Node degree also has an impact on an adversary's view of the protocol's topology and the resources required to reconstruct a full topology map from the local routing views of collaborating nodes.

In this chapter, we identified several design choices we have observed in various ANP2P DHT protocols. Each design choice is described in detail along with its possible implementations. In the next chapter, we will empirically investigate several of these design choices and their impact on performance and anonymity.

CHAPTER 6
EMPIRICAL APPROACH TO DESIGN ANALYSIS

In this chapter, we extended our initial experiments from Chapter 4 and developed an empirical methodology for evaluating the performance and anonymity provided by different design decisions. Our empirical methodology was extended from the work of Borisov [11] to improve the generic adversarial model. Then a study was performed on a subset of the design choices from Chapter 5 using our empirical methodology to better understand their impacts on both performance and anonymity.

We performed an empirical study because ANP2P DHT protocols are often fairly complicated. Previous experiments in Chapter 4 have shown that the interactions between different design choices sometimes have complex relationships. We designed several experiments which allowed us to better understand those interactions and to observe both the performance and the anonymity provided by different design choices.

The remainder of the chapter is separated into three sections. Section 6.1 discusses how we designed and executed our experiments, Section 6.2 discusses how we simulated a generic ANP2P protocol, and Section 6.3 analyzed the data observed in our experiments.

6.1 Experiment Setup

Table 5.1 shows a high-level overview of the design choices that we have studied in this chapter. The main design choices that we focused on are topology type and look ahead. These are the two design choices that we investigated in Section 4.5.6 as potential mitigations for the RTI attack. The remaining design choices used constant values to reduce the complexity of the result analysis. However, the empirical method described in this section can also be applied to the other design decisions as well.

Below is the list of steps that we performed for each design choice in our experiments.

1. Create a random network graph for the given topology type, network size and degree. This network graph is used as the starting state for all the nodes and links in the topology.

2. Randomly pick a subset of the nodes in the topology to be adversary nodes. This uniform choice of nodes does not necessarily represent the best attacker strategy.

3. Pick two random nodes a and b from the topology that are not adversary nodes. Then route a message from a to b. If the message is routed through an adversary's node, then that node will sample the sender's entropy and continue forwarding the message. The message is forwarded so routing performance metrics are accurate. If multiple adversary nodes are encountered while routing, then only the first adversary node will sample the sender's entropy. Repeat

Table 6.1: Empirical Study of Design Choices

Design Choice	**Implementations**
Topology Type	Structured Semi-structured Random
Look Ahead	1 2
Loop Detection	true
Routing Randomness	0%
Hop-to-live	Deterministic
Routing Table Maintenance	None
Backtracking	true
Size	1000
Degree	12
Adversary	2%

this step until a sufficient sample size is obtained. We used a sample size of 100,000 for these set of experiments and parameters.

4. Return to step 1 and repeat. We repeated several times to average out any anomalous properties in the randomly generated topologies.

The first step in our experiments was to create a random topology of a given type. We looked at three different topology types: structured, semi-structured, and random. Each topology type will be described in detail along with how it was generated.

We created a structured topology based on the concept of finger tables in Chord [78] but used undirected links instead of directed. Figure 6.1 shows an example structured topology with 40 nodes. The node at 0.25 is selected, and its peer connections are highlighted in the figure. In order to generate this network, first an ordered ring topology was created by connecting each node to its two closest neighbors. Then shortcut links were added to a node: $1/2^1$, $1/2^2$, ... $1/2^k$ of the address space away. Shortcut links were continually added until each node reached the desired node degree or no more shortcut links could be added. The resulting topology had a routing performance guarantee of $\approx \log(n)$. The other benefit of this topology structure is that we could reuse the greedy routing algorithm with this topology to achieve deterministic routing.

We used small world graphs to represent semi-structured topologies and generated them with the Watts–Strogatz [86] model. The Erdős–Rényi [32] model was used to generate our random topology graphs. The library NetworkX [38] provided the implementations of the Watts-Strogatz and the Erdős–Rényi algorithms.

The routing type used is strongly tied to a topology type. We used the same greedy based routing algorithm for all three of the topology types we evaluated. The greedy algorithm acted like

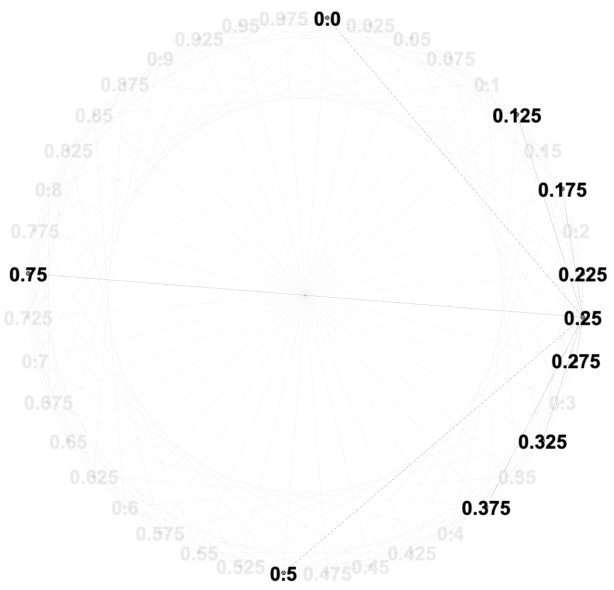

Figure 6.1: Example Structured Topology with 40 nodes

a structured routing algorithm when run on the structured topology, and it provided best effort routing in the semi-structured and random topologies.

The remaining variables all are constant values. The current set of experiments did not model node churn, so we did not need to use routing table maintenance. We used a deterministic HTL value to simplify the analysis process. We have successfully evaluated network sizes up to 10,000 nodes; however, we found that the network size of 1,000 nodes usually maintains a reasonable simulation performance while still accurately representing ANP2P DHT protocol behavior. Future work would be to include system churn and probabilistic HTL as part of our investigation.

6.1.1 Metrics

In this study, we measured the impacts of system design decisions on performance and anonymity. We will first discuss the performance metrics that we collected. Then we will discuss the anonymity metrics along with a detailed explanation of how we calculated the metric.

Performance

The performance metric is straight forward to observe and record. We used the routing path length to represent the performance. The routing path length is the number of hops that a message took to be routed from a sender to a receiver node. This metric includes any failed parts of the routing path where the message must backtrack before continuing on to the destination. The routing path

length metric also abstracts away the underlying network details, so it gives us a performance metric that is decoupled from the underlying network implementation.

Sender Anonymity

Measuring the anonymity provided by ANP2P protocols is a challenging task. The work by Borisov [11] provided an empirical methodology to sample the entropy of ANP2P systems. We extended their work to improve the adversarial model for ANP2P DHT protocols. Our adversarial model calculated two anonymity metrics: *full-sender set* and *top-rank set*.

The full-sender set is calculated by adding all nodes that are reachable within h hops from the adversary. When a message is observed, the adversary uses (or estimates, if HTL is not deterministic) the HTL value h of the message. Then the adversary finds all nodes within h hops from the full network topology and adds them to the full-sender set. If HTL is not used (because of routing path length guarantees in structured topologies, as described in Chapter 5), then the HTL value can still be estimated. The adversary can subtract the maximum routing path length from the remaining path length to calculate the sender's maximum distance from the adversary. We assumed that the adversary has access to the full network topology of the system because at least a sub-set of the topology information must be shared between nodes to enable routing. It is then possible to use multiple colluding adversary nodes to collect the topology sub-sets and reconstruct the full network topology. Determining the resources an adversary would require to reconstruct the full topology is out of scope for this study.

The size of a full-sender set tells us the number of nodes that could potentially be the original sender node, but it is not easy to compare this metric when different experiment configurations are used. To address this, we calculated the entropy provided by the full-sender set and then normalized it. This metric is called *full-sender set entropy*. We assumed that each node in the full-sender set had an equal probability of being the sender. See the entropy equations in Section 2.3.3 for more details on how entropy is calculated. The benefit of the full-sender set entropy metric is that it is easy to calculate and will never miss the original sender, assuming that a sufficient HTL value is used.

The next anonymity metric, top-rank set, utilizes *shared state* and *protocol behavior* information to reduce the size of the possible sender set. See Section 3.2.2 for definitions of shared state and protocol behavior. The top-rank set is calculated from the most likely routing paths using a *Walk-back Ranking Algorithm*. Since the top-rank set contains the most probable sender nodes, we assumed that each node has an equal probability of being the sender. The anonymity metric *top-rank set entropy* can then be calculated from the top-rank set. It is possible to extend our Walk-back Ranking Algorithm to include more than just the top ranked routing paths; however, the computation complexity of the algorithm increases exponentially.

In Section 4.5.1, we described the route prediction model used for the RTI experiments and

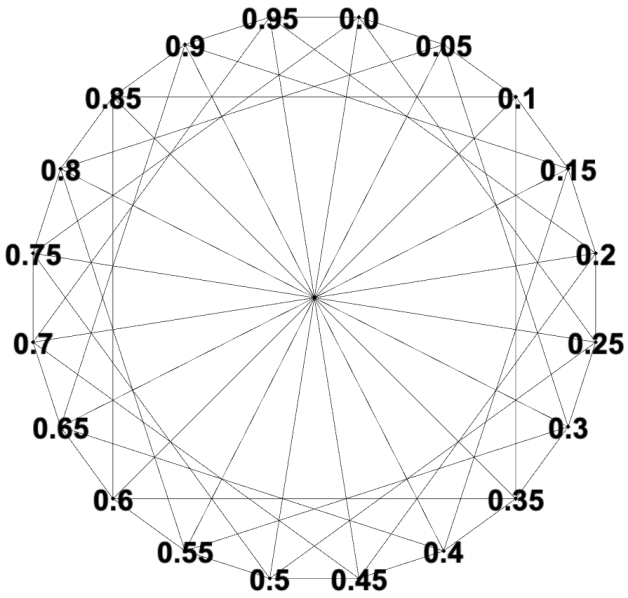

Figure 6.2: Example structured topology with 20 nodes.

analyses; however, scalability was the biggest challenge with continuing to use this model in our empirical study. It took several hours to calculate the full routing path prediction model for an entire network of only a few hundred nodes in the RTI experiments in Section 4.5. Increasing the node count in the network also exponentially increased the computation time. We wanted to scale our node count up to the thousands to better represent real world topologies. This meant we had to create a faster and more efficient model for predicting the original sender of a message. Instead of calculating all possible paths, we only calculate the most likely or top ranked paths.

We will explain how the *Walk-back Ranking Algorithm* works using an example. Figure 6.2 shows a simple network topology of twenty nodes. We know that the protocol is using a deterministic greedy routing algorithm with 1 look ahead. Let's assume the adversary node, a, is node 0.45. Then assume a observes a message that came from node 0.5, and the message has a target address, t, of 0.4. Now call node 0.5 n_{prev} since a knows that it is the previous node in the routing path. Then create an ordered unique list called N_{prev}. N_{prev} is used to track the current routing path that we are building. Below is the list of steps used to calculate the top ranked routing paths.

1. Add n_{prev} to the list N_{prev}.
2. Get all the peers connected to the node n_{prev}. Excluding any nodes that exist in N_{prev}. $P = \{0.0, 0.25, 0.45, 0.55, 0.75\}$ for node 0.5.
3. For each peer p in P, calculate the ordered list of nodes that p would route to when given a

message with the target address of t. Below is an example of ordered peer lists for given P and a target address of 0.4. We can use the *protocol behavior* plus the topology to calculate the routing order for each p in P.

(a) Node 0.0 would produce the ordered list $(0.5, 0.25, \{0.75, 0.05\}, 0.95)$. It is important to note that this ordered node list contains an unordered set in the third position $\{0.75, 0.05\}$. This represents a routing tie. Both of the nodes 0.75 and 0.05 have an equal chance of being routed to.

(b) $0.25 = (\{0.3, 0.5\}, 0.2, 0.75, 0.0)$

(c) $0.45 = (0.4, 0.5, 0.2, 0.7, 0.95)$

(d) $0.55 = (\{0.3, 0.5\}, 0.6, 0.05, 0.8)$

(e) $0.75 = (0.5, 0.25, 0.7, \{0.8, 0.0\})$

4. For each peer p in P, if the first item in the ordered peer list is n_{prev}, then recursively repeat this process by returning to step 1 and assigning p to n_{prev} and passing a copy of N_{prev}.

   (a) In this example, the nodes $\{0.25, 0.55, 0.75\}$ are considered the top candidates for being the previous hop in the routing path. Node 0.45 is excluded because routing to 0.5 was the node's second choice.

   (b) Repeat this process until no new nodes are found as top candidates for routing to the n_{prev} node.

The output of the Walk-back Ranking Algorithm is a tree of possible top ranked routing paths. All the paths lead to the adversary, a, when routing a given message. Figure 6.3 shows what the final routing tree looks like from our example (see Figure 6.8 for a larger example). The tree stops after 2 hops because none of the peers for the leaf nodes would have routed down the represented path as their top choice. The resulting sender set is $TR = \{0.5, 0.25, 0.55, 0.75\}$. We can then calculate a normalized entropy value by assigning equal probability to all the nodes in TR and assigning a 0 probability to the remaining nodes in the network. In this example, we included the node 0.5 in the possible sender set because we did not have a message HTL value to indicate the sender's distance from the adversary, so we included all nodes along the routing paths.

The above process works well for deterministic routing and structured topologies because we know each node will always route to the next closest node. However, it will not work as well for semi-structured topologies and random topologies, which can result in nodes taking their second or third routing choice. We can extend the above process to account for this. First, we perform steps 3 and 4 as they are stated above. If there are no new node candidates found in step 4, then we include nodes that have n_{prev} as their second routing choice.

The top-rank set metric has several benefits over the full-sender set metric. Incorporating *shared state* and *protocol behavior* information allows the metric to reduce the possible set of senders. The

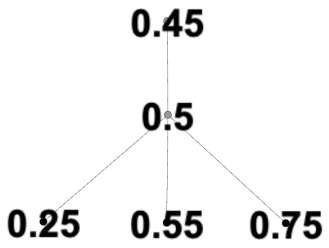

Figure 6.3: Routing Tree of Top Candidates

routing path calculation used here is able to scale significantly over the route prediction model used in Section 4.5.1.

The time to calculate the routing path tree does increase exponentially as the distance between the adversary and a sender increases. We used a maximum distance of 20 hops from an adversary to a sender in our experiments. If an adversary observed a message further than 20 hops from the sender, then the adversary did not attempt to calculate either of the anonymity metrics. We did this because most routing paths fell under 20 hops in our experiments. The only topology that consistently exceeded 20 hops were the random topologies.

Since the top-rank algorithm only has *shared state* to work with, and we only consider the top routing paths, it is possible that the algorithm will not include the actual sender in its reported routing tree. Using the actual state in the simulation environment, we were able to determine if the top-rank set missed the sender or not.

We then used all the above metrics to model the adversary's view of anonymity in the system. If the adversary successfully found the original sender in its top-rank set, then we assigned that as the anonymity metric for the given message. If the adversary missed, then we used the full-sender set as the anonymity metric. This allowed us to account for the accuracy of the top-rank set metric, and to fall back to the next best anonymity metric in the event of a miss.

6.2 Simulation

We used a combination of a hybrid test bed and a custom Freenet simulator in chapter 4 to perform the RTI experiments. The hybrid test bed allowed us to run instrumented Freenet nodes in a completely controlled environment. We then used the custom Freenet simulator, so we could scale to a larger set of nodes. In this chapter, instead of using a custom simulator we extended a popular peer-to-peer simulator called PeerSim.

PeerSim [56] is an open-source P2P protocol simulation framework. It provides the basic P2P simulation of a protocol: create and manage a topology of nodes, provide communication mechanisms, schedule and execute the events generated by each node. We then implemented a generic

DHT protocol [3] on top of the provided simulation. We chose PeerSim because it was well suited to simulating P2P protocols, has been used by other research projects to study P2P DHT protocols, and allowed us to provide more consistent results with similar research projects.

PeerSim does not have any built-in anonymity metrics. As a result, we did not perform the anonymity calculations directly in the simulator. Instead, we calculated the anonymity metrics as part of a post-processing step after running a simulation.

The experiments we ran tested a wide variety of protocol configurations. Some configurations resulted in very poor routing performance, including routing paths that included thousands of nodes in some cases. We found that exceptionally long routing paths significantly degraded the performance of PeerSim. If routing paths became too long, the performance of the established network circuit was too slow to be useful. To avoid this, we added an upper threshold for all routing paths. If a routing path reached a length of 50 hops, the generic DHT simulator stopped routing and marked the message as failed to deliver.

A maximum of 50 hops was chosen because of real world constraints of ANP2P DHT protocols. For example, Freenet uses a maximum HTL value of ≈ 18. We will discuss the impact of this upper bound on our empirical analysis in Section 6.3.1.

6.3 Results

We will discuss the results from our experiments in this section. First, the experiment assumptions will be discussed, and their impact on our data analysis will also be presented. Then we perform a pairwise analysis of both the performance and anonymity impact of the design choice options. A pairwise analysis is done because often one design choice has a direct impact on another design choice.

6.3.1 Experiment Assumptions

We have made several assumptions for our experiments. Each assumption will be explained along with its impact on our analysis.

As explained in Section 6.2, we set a maximum HTL value of 50 hops on our experiments. When a message reached a path length of 50 hops, the simulator marked the message as failed to be delivered and recorded the routing path length of 50. However, we did not exclude failed messages from our analysis. An adversary that observed any of these messages was still able to calculate the potential sender set. The HTL upper bound affected some of our performance analysis by artificially lowering the mean routing path length. We will show the cumulative distribution in addition to the mean value for each performance analysis so that it is evident what portion of the routing paths hit the upper limit.

We did not calculate anonymity metrics if the adversary was more than 20 hops from the sender,

as described in Section 6.1.1. This issue is addressed in the analysis by grouping the experiment observations by the distance between the adversary and the sender. We can then directly compare the anonymity provided at various adversary distances. This is done because a routing path length has a significant impact on the anonymity provided.

All of our experiment configurations used a deterministic HTL value. This means that if an adversary observed a message at a distance of 1 hop from the original sender, then they knew exactly who the sender was, and the anonymity score was 0. We used a deterministic HTL because even with a probabilistic HTL there are still classes of statistical attacks that can determine if the sender is 1 hop away. See the statistical attack on Freenet by Levine, Liberatore, Lynn, and Wright in Section 2.4 for more details. We understand that probabilistic HTLs are an important part of providing anonymity in ANP2P DHT protocols, and we leave this for future work.

The last assumption we made is that 2% of the nodes are malicious. Our focus in these experiments was to know what the anonymity of the protocol looks like with the presence of an adversary, not how many resources an adversary needs to compromise a system. We made the number of adversaries static and kept sending messages between random nodes until we had a good sample of adversary message observations.

In the next subsections, we will compare the performance and anonymity impacts of different topology types and look ahead values. First, we look at the performance and discuss our expected results compared to the observed results. Then, we analyze the anonymity. The anonymity analysis is broken down into two parts. We look at the accuracy of the top-rank set metric proposed in Section 6.1.1, and discuss our observed trends. Finally, the mean and minimum anonymity by the distance between an adversary and a sender is discussed.

6.3.2 Performance

Figure 6.4 shows the cumulative distribution and Figure 6.5 shows the box plots for routing path lengths grouped by topology type and look ahead. We included the box plots to better visualize the change in how data points are grouped as design choice parameters are varied. First, let's look at the random topology with 1 look ahead. In Figure 6.4, we can see that 41.5% of the routing paths in this configuration have a length of 49 hops or less. See Table 6.2 for the exact numbers. This means the remaining 58.5% of paths hit the maximum upper bound of 50 hops, so we can conclude that this configuration does not provide adequate performance. This was expected since random topologies have no explicit organization that can be utilized to find nodes efficiently.

Next, we will look at the random topology with a look ahead of 2. We can see that look ahead has a significant impact on the routing performance, and only small portion of the traffic, 2%, is still hitting the maximum HTL of 50. Table 6.2 shows the exact percentage of messages that have a routing path of 50 or greater. Here we can see that a random topology used with a 2 look ahead has a mean routing performance much closer to some small world and structured configurations.

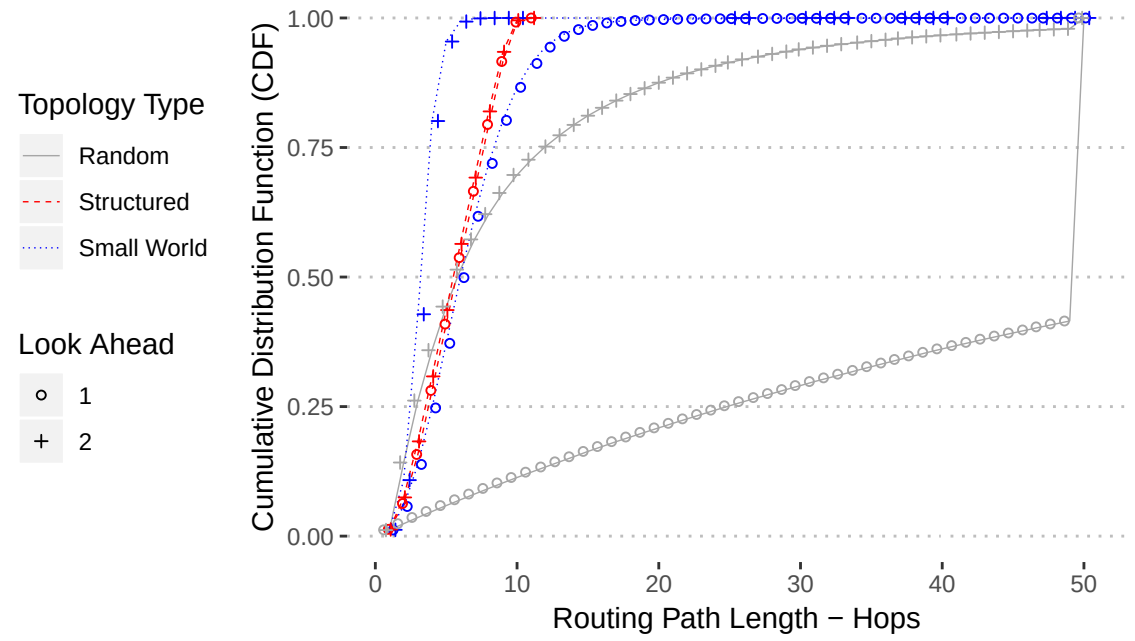

Figure 6.4: CDF of Routing Path Lengths by Topology Type and Look Ahead

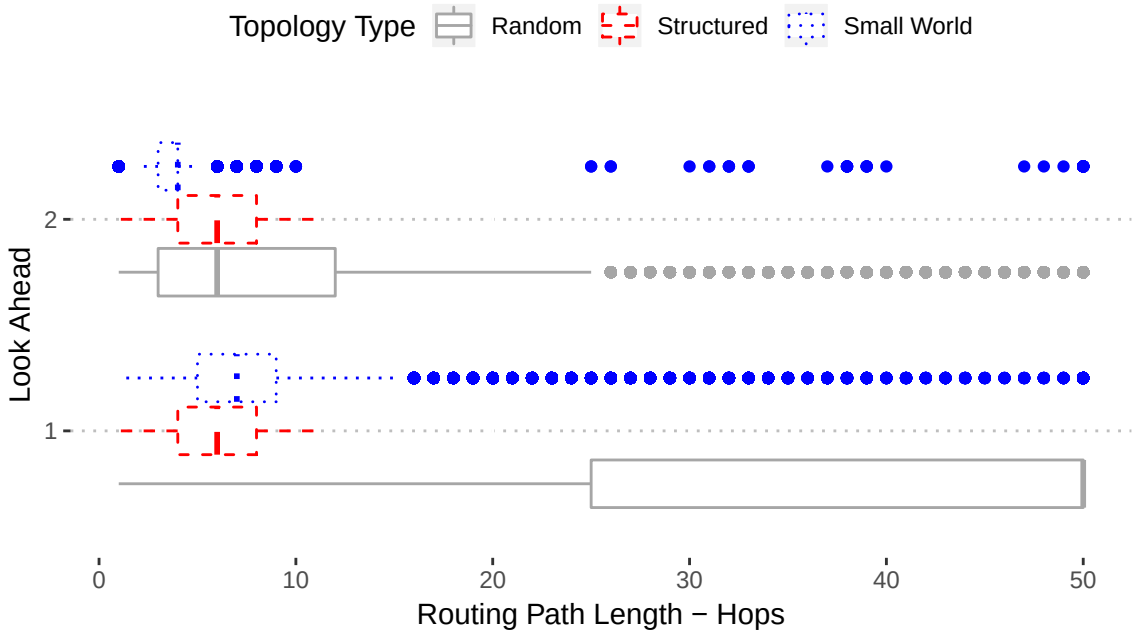

Figure 6.5: Box Plot of Routing Path Lengths by Topology Type and Look Ahead

Table 6.2: Mean Routing Path Length and Percentage of Path Lengths 50 or Greater

Topology Type	**Look Ahead**	**Mean Length**	**Paths $\geq$ 50**
Random	1	38.3	58.5%
Random	2	10.1	2.09%
Small World	1	6.93	0.0235%
Small World	2	3.71	0.00225%
Structured	1	6.17	0%
Structured	2	5.98	0%

However, it still has a long tail, and there are still a large number of routing path lengths that are considered outliers.

When viewing Figure 6.4 and 6.5, the distributions for structured topologies using 1 look ahead and 2 look ahead are difficult to distinguish. A Kolmogorov-Smirnov Test was performed to confirm that the differences between these two configurations were statistically significant. The structured topology performance was slightly improved after increasing the look ahead from 1 to 2. The performance improvement was expected because 2 look ahead allows the structured routing algorithm to discover more efficient routing paths. If a more efficient route was not discovered, the same route used during 1 look ahead routing was used. The goal of the experiments was not to find the most performant structured topology and routing algorithm, but to study the impacts of the upper path length limit and the deterministic routing paths.

The small world topology performance was significantly impacted by look ahead. The mean, the number of routing paths greater or equal to 50 and the size of the interquartile range (IQR) in Figure 6.5 are all reduced. However, we can see that a small percentage of the messages, 0.00225%, from Table 6.2 are still being routed at least 50 hops. Further analysis into those routing paths determined that it was caused by the semi-random nature of the small world topologies. The topologies use a best-effort algorithm to construct and maintain their peer relationships. Since it is best effort, not all nodes are guaranteed to have the correct proportion of close peers compared with far-away peers that small world topologies expect. For example, the small world with 2 look ahead configuration in Figure 6.5 has several outliers, all caused by trying to route through two nodes that did not have proper peer distance proportions. The same is also true for the small world with 1 look ahead configuration; however, this configuration is more sensitive to nodes that do not have the correct small world ratio. Small world topologies on average have good performance; however, they cannot guarantee performant delivery for all messages.

So far, we have looked at the performance impact of look ahead and topology type. In all three topology types, increasing the look ahead did provide a statistically significant distribution change, and resulted in decreased routing path lengths. The performance improvement amount is greatly impacted by the topology type. The topology type also has a significant impact on the distribution of routing path lengths. Structured topologies have a guaranteed upper bound on routing path

lengths, and do not have any outliers in Figure 6.5 or routing paths greater or equal to 50 hops in Table 6.2. The small world and random topologies do have outlier path lengths and paths greater or equal to 50 hops because of the random nature in how they are constructed. The determination for which combination of design choices is ideal will depend on system requirements and what is considered acceptable for a given application.

6.3.3 Anonymity

In this section, we discuss the anonymity provided in our various experiment configurations. First, we looked at the accuracy of our top-rank set metric. Then we investigated the mean, minimum, and adjusted top-rank set metric.

Top-Rank Set Accuracy

We used the full-sender set and top-rank set to evaluate the impacts of design choices on anonymity. However, before examining entropy, the accuracy of the selected metrics needs to be examined. Accuracy is measured as either a *hit* or *miss*. A hit means that the sender node of a message was found in the given set, and a miss means the sender node was not found. We used the global state in the simulation environment to measure the accuracy of the metric. The full-sender set had an accuracy rate of 100%. This means that the full-sender set always included the correct sender node. Next, we analyzed the accuracy of the top-rank set.

The top-rank set accuracy can be found in the Figure 6.6. Accuracy in this figure is represented as the percentage of adversary predictions that correctly identified the sender node of a given message. Figure 6.6 only shows the entropy for messages intercepted by an adversary's node. In the remainder of this chapter, we build on the top-rank set metric to provide various views of the system's anonymity. First, the top-rank set accuracy for random topologies with both 1 and 2 look ahead configurations will be analyzed. Then the structured topologies will be analyzed, followed by the small world.

First, we analyzed the top-rank set accuracy of 1 and 2 look ahead for random topologies. See Figure 6.6. We found that the accuracy with random topologies quickly declined relative to the other topology types as the routing distance between the adversary and the sender increased. This was expected behavior because the Walk-back Ranking Algorithm was designed to take advantage of the structured nature of topologies, and random topologies inherently lack structure. Our experiment data suggested that the top-rank set metric can be used for random topologies if the distance between an adversary and a sender is less than or equal to the look ahead value + 2 hops. In our experiments 18.97% of the intercepted messages are within the look ahead + 2 hops in random topologies with 1 look ahead and 47.19% in random topologies with 2 look ahead. In conclusion, the top-rank set metric is viable for random topologies if the adversary is close to the sender.

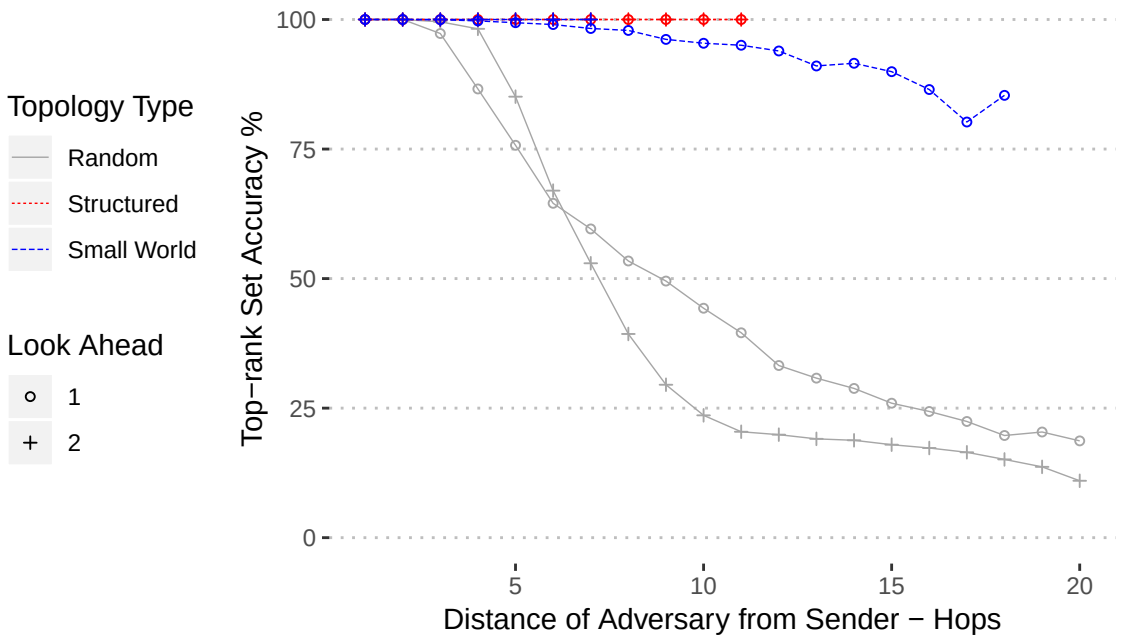

Figure 6.6: Accuracy of Top-rank Set by Topology Type and Look Ahead

Next, we analyzed the top-rank set accuracy for small world topologies with 2 look ahead and structured topologies with both 1 and 2 look ahead. All of these configurations had a 100% accuracy rate for all observed distances between the adversary and the sender. This was expected behavior for both of the structured topology configurations because of the topology's invariance and deterministic routing. Our experiment data also showed a 100% accuracy rate for all small world topologies with 2 look ahead. We hypothesize that this is because of the relatively short routing path lengths and features of the 2 look ahead mechanism. One of those features is if the look ahead value is greater than 1, then the look ahead mechanism can reduce local minimums along the routing path by increasing each node's visibility of the full topology. We will explore this topic more in the next paragraph when discussing the small world topologies with 1 look ahead.

The last configuration in Figure 6.6 is small world topologies with 1 look ahead. The accuracy of this configuration gradually declined as the distance between the adversary and the sender nodes increased. This was expected behavior because of the semi-random nature of how the small world topologies are created and maintained. We sampled the top-rank sets that missed the sender node in order to explain why the accuracy decreased as the distance increased. The common element that we found was the messages were routed through a part of the topology that did not conform to the desired small world peer distance ratio. This resulted in the message getting stuck in a local minimum and having to backtrack. In our experiment data, the small world topologies with 1 look ahead had an accuracy rate of 95.42% at a distance of 10 hops. This is still a relatively

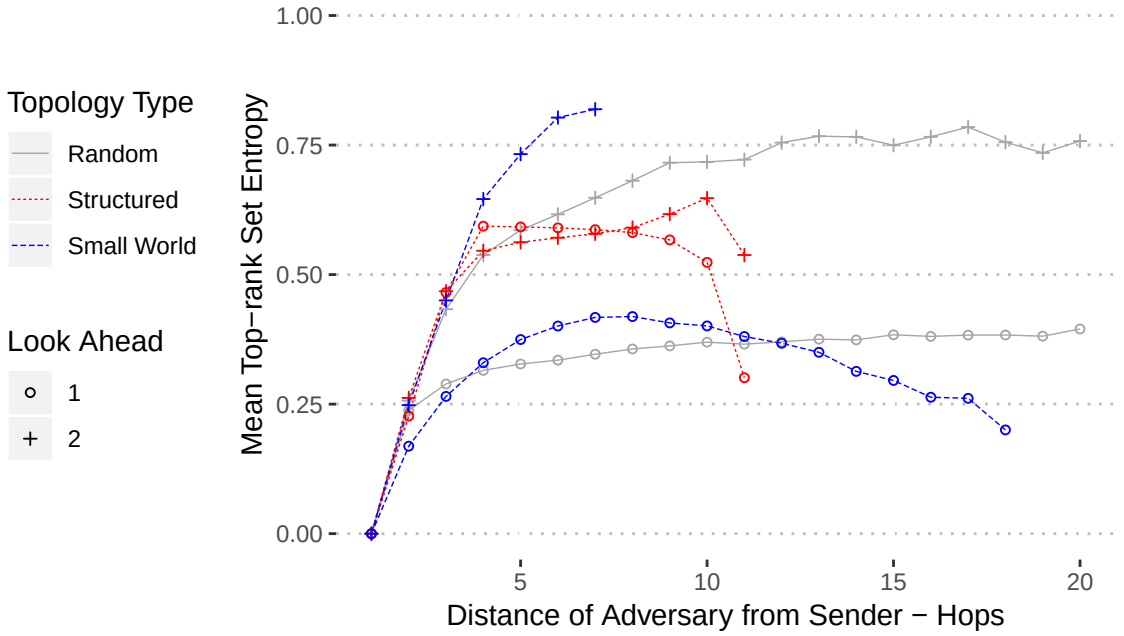

Figure 6.7: Mean Top-rank Set Entropy by Topology Type and Look Ahead

high accuracy rate, and it covered 96.15% of the intercepted messages. In conclusion, the top-rank set metric has a high accuracy rate for a vast majority of the intercepted messages for small world topologies with 1 look ahead. An adversary could also take steps to reduce the distance between themselves and the sender to increase their top-rank set accuracy.

Mean Top-Rank Set Entropy

After we analyzed the accuracy of the top-rank set, we then investigated the mean top-rank set entropy. Our initial analysis only included the top-rank sets that successfully *hit* their sender node. Later in this chapter we will expand on this analysis to include the *missed* top-rank set.

First, we investigated the random topologies in Figure 6.7. Our experiment data showed that increasing the look ahead value from 1 to 2 also increased the mean top-rank set entropy. The increased entropy was a result of the look ahead mechanism increasing the number of potential routing paths in a given topology. The increased routing path choices then resulted in more routing path ties and larger top-rank set sizes. It is also important to note that Figure 6.7 only shows the top-rank sets that *hit* their sender nodes. The random topology configurations in Figure 6.6 can be cross-referenced with Figure 6.7 to determine what percentage of the intercepted messages are being represented. We also found that when the distance increased between the adversary and the sender nodes, the top-rank set entropy for random topologies also increased. Due to experiment

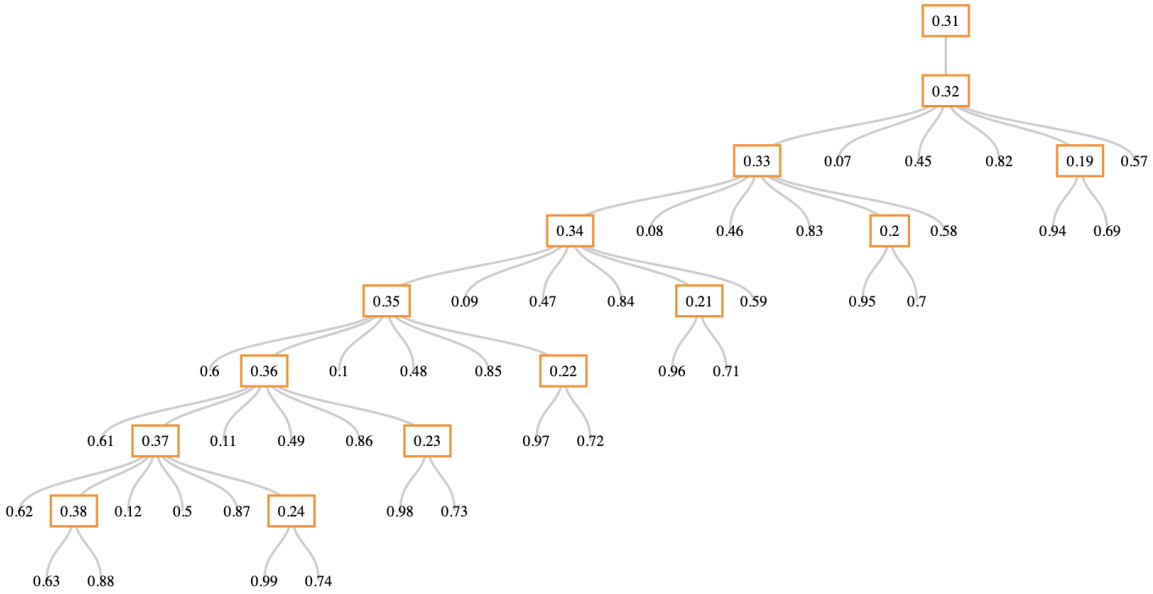

Figure 6.8: Example Top Ranked Routing Path Tree with Adversary Distance of 8 hops

and protocol performance constraints we did not calculate the top-rank set for distances greater than 20 hops so we are unsure if the random topologies will eventually plateau and start declining like the small world and structured topologies.

The next experiment configuration we analyzed was the structured topologies with 1 look ahead in Figure 6.7. Our experiment data showed that the mean top-rank set entropy peaked at an adversary distance of 4 hops. After 4 hops the mean entropy then gradually declined. This continues until (but not including) the maximum hop distance. Then at the maximum hop distance there is a significant drop in entropy. In order to help explain this behavior, we created a simplified (fewer nodes) structured topology example.

We created an example structured topology with 100 nodes and a degree of 7. Figure 6.9 shows the mean top-rank set entropy for this example configuration. Notice that the example topology in Figure 6.9 displays similar properties as the random topologies with 1 look ahead in Figure 6.7. Figure 6.8 shows an example top ranked routing path tree generated by our Walk-back Ranking Algorithm for the 100 node example topology. The root node of the tree is the adversary in this instance, and the message target address was 0.31. Each level in the tree represents the top-rank set for a given hop distance from the adversary up to the maximum hop distance of 8. For example, the top-rank set for a sender 5 hops away is $\{0.6, 0.36, 0.1, 0.48, 0.85, 0.22, 0.96, 0.71\}$ according to Figure 6.8.

We found that there were two main competing factors affecting the mean top-rank set entropy

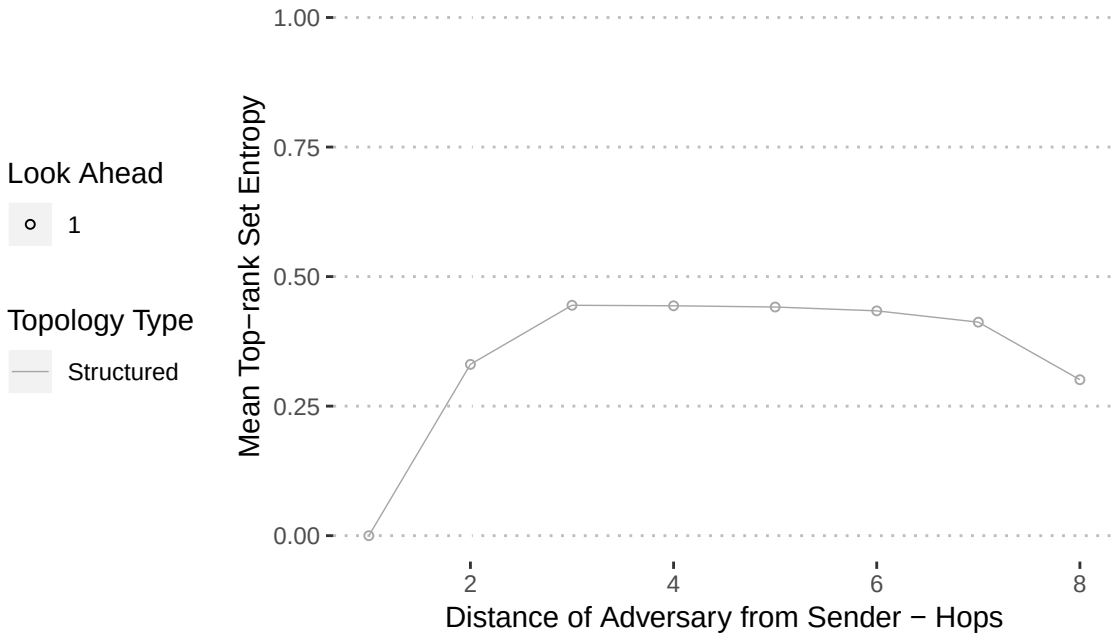

Figure 6.9: Example Mean Top-rank Set Entropy for Example 100 node Structured Topology

in Figure 6.9. The first factor was the increasing adversary distance. As the distance between the adversary and sender increased, so did the number of nodes that could potentially route to the adversary. In Figure 6.8, we can see that the top-rank set size increased for all hop distances less than or equal to 3. However, then the number of nodes in the top-rank set stopped growing. This was caused by the second factor, which is routing path restrictions. The topology and routing algorithm working together create a set of restrictions on what paths a message will take based on the message's position in the topology and its target address. In Figure 6.8, we can see a repeating pattern for how the next level of top-rank set nodes are picked. This continues until the maximum hop distance is reached. At the maximum path length, our structured topology always created a top-rank sender set that was half the size of the maximum top-rank set size. This was not a known property when we designed the structured topology used in our experiments. The gradual decline in entropy after the peak was also explained by this property. When the adversary distance increased, the probability that the adversary would intercept a message with the maximum routing path length also increased. This behavior caused the overall mean top-rank set entropy to decline after the peak in Figure 6.9 (and Figure 6.7).

One of the benefits of using our empirical methodology is that the adversarial model is capable of modeling the complex interactions between the protocol's various components. We used the output of our adversarial model to identify a previously unknown property of our structured topologies (Section 6.1) that caused a degradation in the mean entropy provided by the system. We leave

fixing (or replacing) the structured topology used in our experiments as future work.

The next experiment configuration we analyzed was the structured topologies with 2 look ahead in Figure 6.7. The 2 look ahead mechanism not only increased the possible routing paths, but it also created some short-cut links in the topology. This resulted in slightly better performance (Figure 6.5) versus the 1 look ahead. However, it also added variations to the structured nature of the topology. Figure 6.10 better shows the impact of this added variation. The increased look ahead not only increased the maximum entropy, but it also caused a lower minimum entropy. The last behavior we analyzed for the structured topologies with 2 look ahead was the drop at the end. Structured topologies with both the 1 and 2 look ahead had a drop in their entropy at the maximum adversary distance because of the half-sized top-rank set issue when message routes are at the maximum length. Based on our experiment data, we found that the 2 look ahead mechanism did not provide enough of a performance increase to justify the reduction in entropy.

Finally, we looked at the small world topologies in Figure 6.7. The small world topologies with 1 look ahead behaved similar to the structured topologies with 1 look ahead. The entropy peaked at an adversary distance of 8, and then it gradually declined. Because of the complex protocol behavior and semi-randomness of the small world topologies, we leave quantifying why the shift occurs at the given location for future work. Next, we found that the small world topologies with 2 look ahead offer the highest mean top-rank set entropy of our experiment configurations. Because of the clustering coefficient in small world graphs, increasing the look ahead greatly increases the number of top ranked routing ties. The increased routing ties then caused an increase in the top-rank set size.

Minimum Top-Rank Set Entropy

After we examined the average entropy, we looked at the worst-case entropy. The minimum top-rank set entropy in Figure 6.10 shows the worst-case anonymity. The experiment configurations (small world with 1 look ahead), (small world with 2 look ahead), and (random with 1 look ahead) all have a minimum entropy value of 0. This means that at each of the adversary distances, there was at least one routing path in which the adversary was able to reduce the top-rank set down to a single sender node and hit the correct sender. We briefly investigated this phenomena since it was happening at least once for every adversary distance in small world topologies. The entropy value of 0 was caused by the node's placements in the topology. Because of the node's location relative to its neighbors, the adversarial model we used was able to identify a single node as the only likely sender node. Further investigation is required to determine what properties allow this unique identification of a sender node from any adversary distance in small world and some random topologies. Protocol designers could then use this information to create controls that could prevent these vulnerable sub-graphs from forming in their topology.

In the last experiment configuration, we analyzed the minimum entropy for structured topolo-

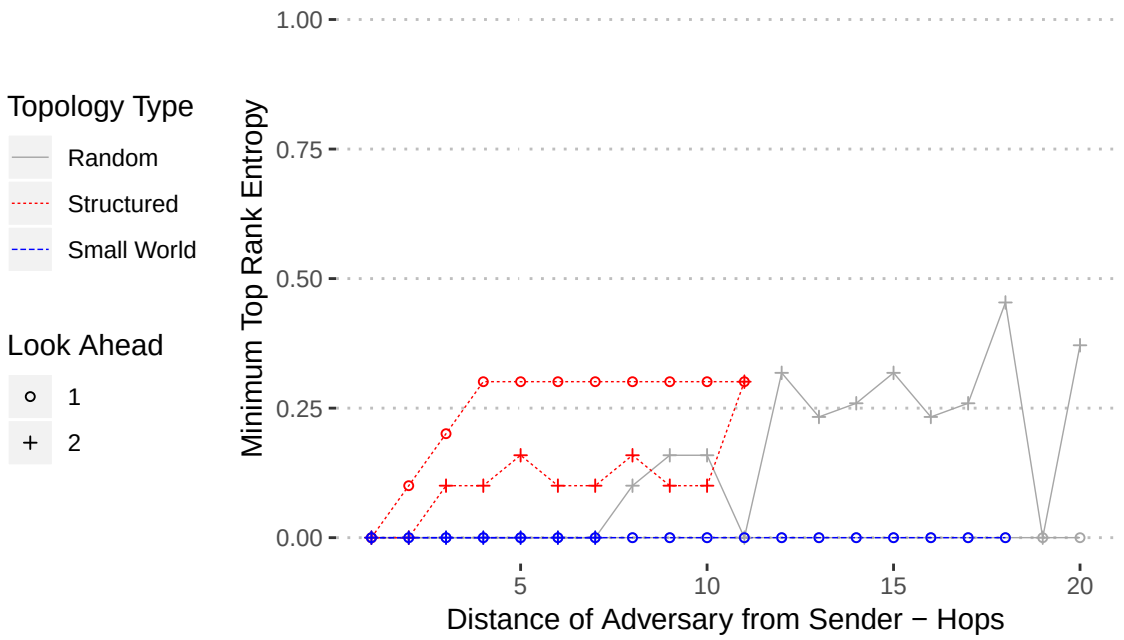

Figure 6.10: Minimum Top-rank Set Entropy by Topology Type and Look Ahead

gies. The 1 look ahead configuration has a very well-defined set of minimum entropy values. This is caused by all nodes having the exact same peer distance distribution, and it guarantees routing path ties. The decrease in minimum values for the 2 look ahead configuration is because of the unique shortcut paths created by the look ahead. However, the last data points at the maximum routing path length are equal for 1 and 2 look ahead because no shortcut paths were taken for 2 look ahead. If a shortcut path had been taken, then the routing path length wouldn't be the maximum value.

Adjusted Top-Rank Set Entropy

The mean adjusted entropy is used to account for the accuracy of the top-rank set metric. If the adversary's top-rank set successfully hit the sender node, then the top-rank set entropy value will be used for that observation of the system. If the adversary's top-rank set misses the sender node, then the full-sender set will be used to calculate the entropy. The full-sender set is a more robust metric that is much less likely to miss the sender node. The robustness of the metric comes at the cost of an increased potential sender set size. Figure 6.11 shows the mean adjust entropy for our experiments.

Random topology configurations are most impacted by adjusting the entropy value. In Figure 6.11, the anonymity of random topologies was greatly increased by adjusting the entropy. This is

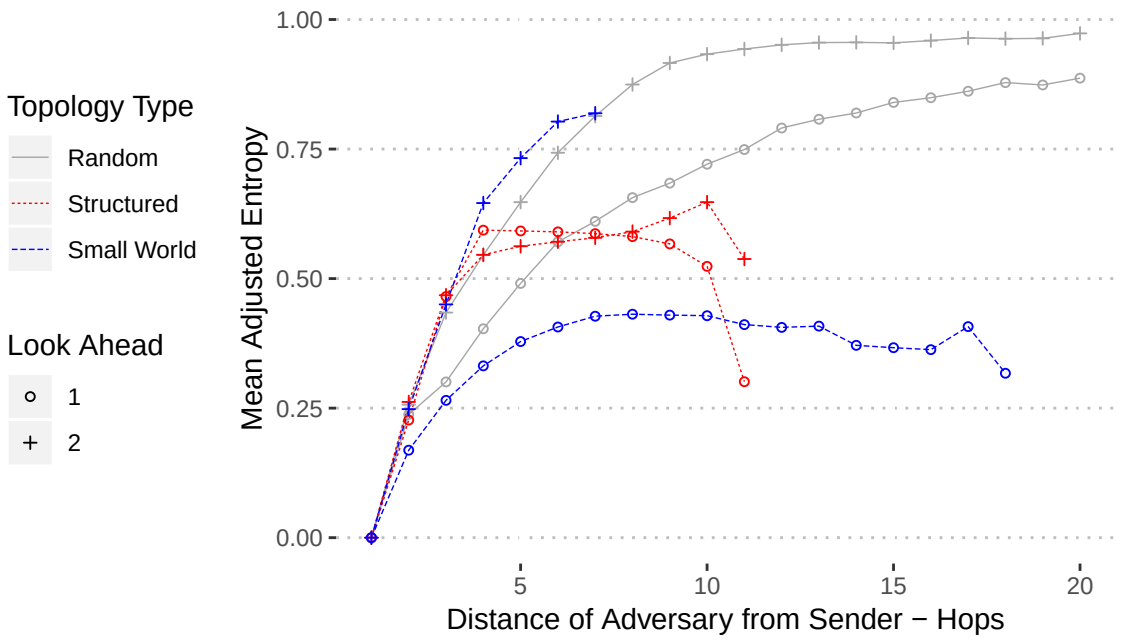

Figure 6.11: Adjusted Mean Top-rank Set Entropy by Topology Type and Look Ahead

because of the low accuracy rate of using the top-rank methodology on random topologies. We can use the mean adjusted entropy to get a better representation of the average entropy provided by a given protocol configuration.

6.4 Summary

In this chapter, we defined an empirical methodology for evaluating the performance and anonymity provided by design decisions. The methodology included a generic adversarial model that could calculate the full-sender set and top-rank set for a given intercepted message. The adversary model used the shared state, protocol behavior, and a walk-back ranking algorithm to reduce sender anonymity. Next, we described how PeerSim was used to run our experiments.

We then ran the empirical methodology on a subset of the design choices defined in Chapter 5. The routing performance and observed anonymity metrics for topology type and look ahead were evaluated. Those metrics allowed us to compare various design choices and their impact on performance and anonymity.

Alternatively, these metrics can also be used to identify weaknesses in protocol design. For example, the small world topology with 2 look ahead has better mean performance than structured with 1 look ahead in our experiments, but the small world topology also has a lower (zero) minimum top-rank set entropy. If it were possible to detect the network sub-graphs causing zero entropy and

prevent them from forming in the topology, we could potentially increase the entropy and improve the anonymity. Another example of this was the issue we identified in our structured topologies. If the routing path length is at the maximum length, then the top-rank set is reduced by half. The work described in this chapter can be used to identify these and other design flaws in ANP2P DHT protocol specifications.

Based on the observations from our experiments, we can conclude that structured 1 look ahead configuration provides the best trade-off between performance and anonymity. This configuration has the best worst-case scenario of the configurations that we analyzed. However, different systems may have other considerations or requirements, and the methodology described in this chapter can be used to examine other dimensions of the data.

CHAPTER 7
CONCLUSIONS

7.1 Summary of Contributions

We performed an in-depth analysis of the ANP2P DHT protocols Freenet and GNUnet. The analysis resulted in the identification of three vulnerabilities. The first vulnerability that we found was the Traceback attack in Freenet. This attack enables an adversary to identify the subset of nodes that had routed a given message using a direct peer connection. A direct peer connection is required to perform the Traceback attack, so we then developed the Routing Table Insertion (RTI) attack so adversaries can force their nodes into a victim's routing table. The RTI attack takes advantage of the deterministic nature of Freenet's peer replacement policy. We then presented two different mitigations to the RTI attack. The first mitigation was to add more randomness to the routing algorithm, and the second was to use a Look Ahead Hint. The performance and anonymity impact of each mitigation was then evaluated. The last vulnerability that was identified during our analysis was the bloom filter GNUnet uses for loop detection. Each message sent in GNUnet includes a bloom filter that contains an entry for every node that routed the message. We theorized that the bloom filter can be used to perform an attack very similar to the Traceback attack.

Next, we developed an empirical methodology to evaluate the performance and anonymity provided by design decisions. This methodology included a novel approach for modeling the adversary's capabilities using the shared state and protocol behavior. The adversary model uses a Walk-back Ranking Algorithm to calculate the possible routing paths. We used this adversarial model to construct potential sender sets and evaluate the anonymity provided by the protocol. We also only used the top-ranked routing paths, and this allowed us to scale the approach to large topologies and perform the calculation in near real time.

The empirical methodology was run on a subset of the design decisions that we have encountered in ANP2P DHT protocols. The routing performance and observed anonymity metrics for topology type and look ahead were evaluated. Those metrics allowed us to compare various design choices and their impact on performance and anonymity.

Protocol designers can use our empirical methodology to evaluate anonymity and performance of their protocols, and to identify the existence of network sub-graphs that degrade anonymity. Once these sub-graphs have been identified, protocol designers can create appropriate controls to prevent their formation.

7.2 Future Work

For future work, we would like to evaluate the effectiveness of the bloom filter attack on GNUnet. It should be possible to reliably identify all the nodes that have routed a given message; however, the attack requires that the adversary collects the public ids for all nodes.

Additionally, we only ran our empirical methodology on a small subset of the available ANP2P design decisions. The next steps would be to perform the analysis on the remaining design decisions in Chapter 5 and the RTI attack mitigations proposed in Section 4.5.6. In addition, we would also like to consider different node degree distributions in our experiments.

We have demonstrated that our empirical methodology can work on ANP2P DHT protocols using simulation. The next steps are to evaluate our empirical methodology on a real-world protocol. Freenet would be an ideal candidate to further this study.

There are also several experiment simplifications that we would like to explore further. No network churn is present in simulations that we used; however, churn is a normal part of ANP2P DHT operations. Most nodes will only join and participate in the network for a limited amount of time before leaving. We would also like to model probabilistic HTL values in our adversary model as future work.

During the empirical study we found certain unique routing paths that resulted in our adversarial model being able to correctly reduce the potential sender set to a single node. We would like to explore what properties in the small world and random topologies resulted in these unique routing paths.

APPENDIX A
FREENET PROTOCOL

A.1 CHK Request Data

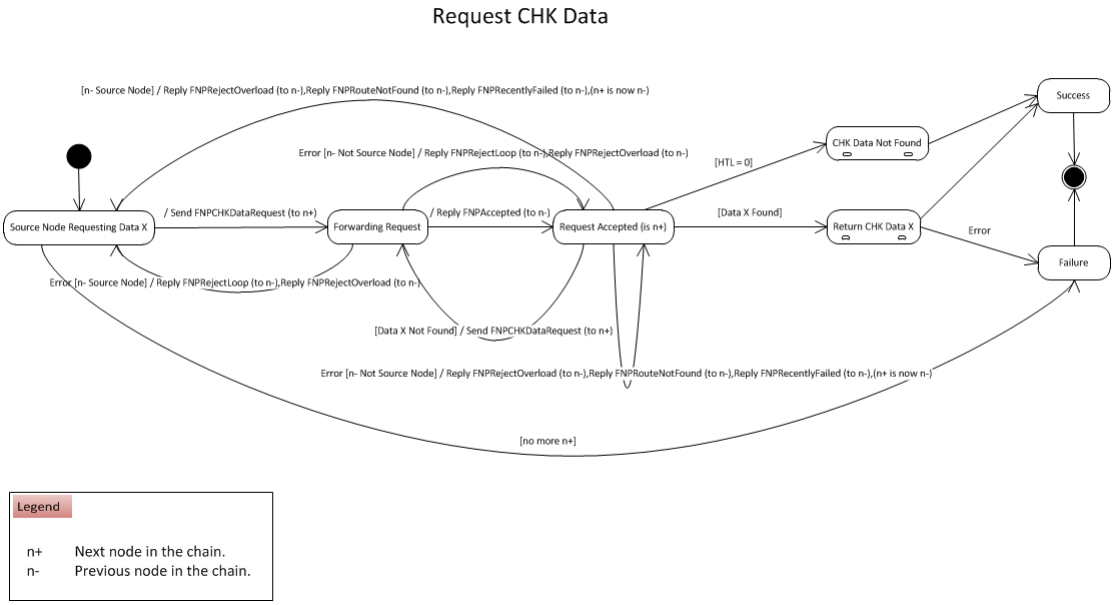

Figure A.1: State transition diagram: request CHK data.

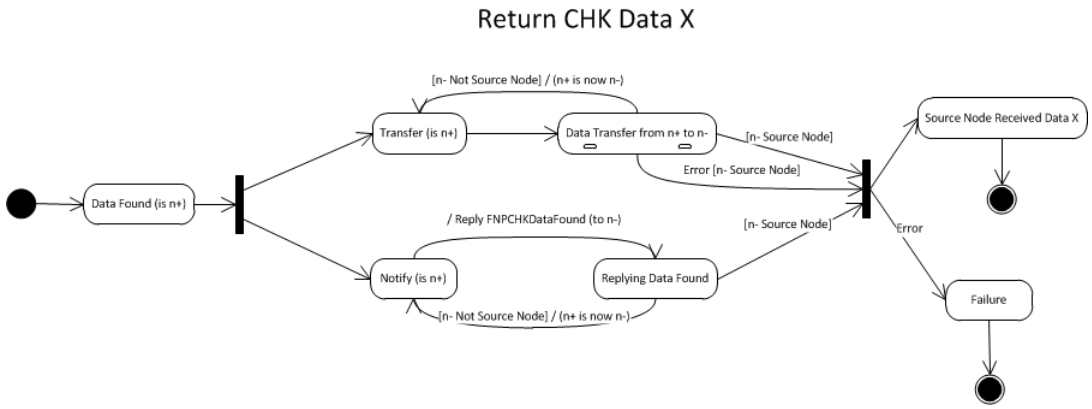

Figure A.2: Sub state transition diagram: return CHK data X.

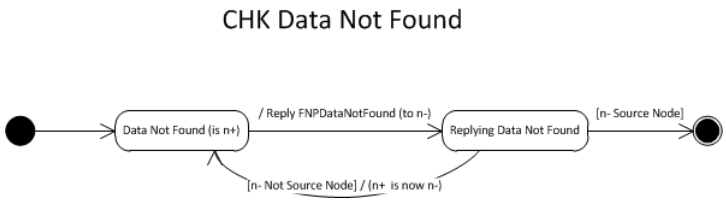

Figure A.3: Sub state transition diagram: CHK data not found.

BIBLIOGRAPHY

[1] Ludovic Barman, Mahdi Zamani, Italo Dacosta, Joan Feigenbaum, Bryan Ford, Jean-Pierre Hubaux, and David Wolinsky. Prifi: A low-latency and tracking-resistant protocol for local-area anonymous communication. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, pages 181–184. ACM, 2016.

[2] T Baumeister, Yingfei Dong, Zhenhai Duan, and Guanyu Tian. A Routing Table Insertion (RTI) Attack on Freenet. In *2012 ASE/IEEE International Conference on Cyber Security (CyberSecurity)*, pages 8–15, Washington D.C., USA, December 2012.

[3] Todd Baumeister. PeerSim Generic DHT Source Code. https://github.com/tbaumeist/peersim-generic-DHT, 2018.

[4] Todd Baumeister, Yingfei Dong, Zhenhai Duan, and Guanyu Tian. Routing Table Insertion Attack and Route Prediction on Freenet: Technical Report. Technical report, Department of Information and Computer Sciences, University of Hawaii, Honolulu, Hawaii, 2012.

[5] Todd Baumeister, Yingfei Dong, Guanyu Tian, and Zhenhai Duan. Look-Ahead Hint: Reduce Routing Information Sharing To Mitigate Routing based Attacks. Technical report, Department of Information and Computer Sciences, University of Hawaii, Honolulu, Hawaii, 2013.

[6] Todd Baumeister, Yingfei Dong, Guanyu Tian, and Zhenhai Duan. Using Randomized Routing to Counter Routing Table Insertion Attack on Freenet. In *IEEE Globecom 2013 - Communication and Information System Security Symposium (GC13 CISS)*, Atlanta, Georigia, December 2013.

[7] Ingmar Baumgart and Sebastian Mies. S/Kademlia: A practicable approach towards secure key-based routing. In *Parallel and Distributed Systems, 2007 International Conference on*, volume 2, pages 1–8. IEEE, 2007.

[8] Krista Bennett, Krista Bennett, Christian Grothoff, Tzvetan Horozov, Ioana Patrascu, and Tiberiu Stef. GNUnet - A truly anonymous networking infrastructure. *in: Proc. Privacy Enhancing Technologies Workshop*, 2002.

[9] Krista Bennett, Tiberius Stef, Christian Grothoff, Tzvetan Horozov, and Ioana Patrascu. The GNet Whitepaper. *Purdue University*, 06/2002 2002.

[10] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In *Designing Privacy Enhancing Technologies*, pages 115–129. Springer, 2001.

[11] Nikita Borisov. *Anonymous routing in structured peer-to-peer overlays*. PhD thesis, Citeseer, 2005.

[12] Konstantinos Chatzikokolakis, Tom Chothia, and Apratim Guha. Statistical measurement of information leakage. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 390–404. Springer, 2010.

[13] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Anonymity protocols as noisy channels. *Information and Computation*, 206(2):378–401, 2008.

[14] D Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology*, 1(1):65–75, 1988.

[15] D L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[16] Han Chen and Pasquale Malacaria. Quantifying maximal loss of anonymity in protocols. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 206–217. ACM, 2009.

[17] I Clarke, S G Miller, T W Hong, O Sandberg, and B Wiley. Protecting free expression online with Freenet. *Internet Computing, IEEE*, 6(1):40–49, January 2002.

[18] I Clarke and Others. A distributed decentralised information storage and retrieval system. *Undergraduate Thesis*, 1999.

[19] Ian Clarke, Oskar Sandberg, Brandon Wiley, and TheodoreW. Hong. Freenet: A distributed anonymous information storage and retrieval system. In Hannes Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *Lecture Notes in Computer Science*, pages 46–66. Springer Berlin Heidelberg, 2001.

[20] Tyson Condie, Varun Kacholia, Sriram Sank, Joseph M Hellerstein, and Petros Maniatis. Induced Churn as Shelter from Routing-Table Poisoning. In *NDSS*, 2006.

[21] G Danezis, R Dingledine, and N Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 2–15. IEEE, 2003.

[22] George Danezis and Prateek Mittal. SybilInfer: Detecting Sybil Nodes using Social Networks. In *NDSS*, 2009.

[23] George Danezis and Andrei Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In *International Workshop on Information Hiding*, pages 293–308. Springer, 2004.

[24] Anupam Das, Nikita Borisov, Prateek Mittal, and Matthew Caesar. Re 3: relay reliability reputation for anonymity systems. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 63–74. ACM, 2014.

[25] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In *Privacy Enhancing Technologies*, pages 184–188. Springer, 2003.

[26] R Dingledine, M Freedman, and D Molnar. The free haven project: Distributed anonymous storage service. In *Designing Privacy Enhancing Technologies*, pages 67–95. Springer, 2001.

[27] R Dingledine, N Mathewson, and P Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[28] Shlomi Dolev and Rafail Ostrobsky. Xor-trees for efficient anonymous multicast and reception. *ACM Transactions on Information and System Security (TISSEC)*, 3(2):63–84, 2000.

[29] J Douceur. The sybil attack. *Peer-to-peer Systems*, pages 251–260, 2002.

[30] S. Dougherty, X. Luo, B. Massop, M. Nyhus, D. Roden, M. Toseland, and M. Treydte. Freenet Source Code. https://github.com/freenet/fred-staging, 2013.

[31] Steve Dougherty. Freenet Statistics. https://www.asksteved.com/stats/, August 2017.

[32] P. Erdős and A. Rényi. On random graphs I. *Publ Math Debrecen*, 1959.

[33] Nathan S Evans and Christian Grothoff. R5n: Randomized recursive routing for restricted-route networks. In *Network and System Security (NSS), 2011 5th International Conference on*, pages 316–321. IEEE, 2011.

[34] Joan Feigenbaum, Aaron Johnson, and Paul Syverson. Probabilistic analysis of onion routing in a black-box model. *ACM Transactions on Information and System Security (TISSEC)*, 15(3):14, 2012.

[35] M J Freedman and R Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 193–206. ACM, 2002.

[36] Yong Guan, Xinwen Fu, Riccardo Bettati, and Wei Zhao. A quantitative analysis of anonymous communications. *Reliability, IEEE Transactions on*, 53(1):103–115, 2004.

[37] Mesut Gunes, Udo Sorges, and Imed Bouazizi. ARA-the ant-colony based routing algorithm for MANETs. In *Parallel Processing Workshops, 2002. Proceedings. International Conference on*, pages 79–85. IEEE, 2002.

[38] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[39] Jinsong Han and Yunhao Liu. Rumor riding: Anonymizing unstructured peer-to-peer systems. In *Network Protocols, 2006. ICNP'06. Proceedings of the 2006 14th IEEE International Conference on*, pages 22–31. IEEE, 2006.

[40] Jinsong Han and Yunhao Liu. Mutual anonymity for mobile p2p systems. *Parallel and Distributed Systems, IEEE Transactions on*, 19(8):1009–1019, 2008.

[41] Jinsong Han, Yunhao Liu, Li Xiao, Renyi Xiao, and Lionel M Ni. A mutual anonymous peer-to-peer protocol design. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 68a—-68a. IEEE, 2005.

[42] Steven Hazel, Brandon Wiley, and O. Wiley. Achord: A variant of the chord lookup service for use in censorship resistant peer-to. In *... 1st International Workshop on Peer-to-Peer Systems (...* . Peer Publishing Systems, Proc. Second Intâ€™l Conf. on Peer to Peer Computing, 2002.

[43] Tomas Isdal, Michael Piatek, Arvind Krishnamurthy, and Thomas Anderson. Privacy-preserving P2P data sharing with OneSwarm. *ACM SIGCOMM Computer Communication Review*, 2012.

[44] jrandom (Pseudonym). Invisible internet project (i2p) project overview. Design document, August 2003.

[45] M Frans Kaashoek and David R Karger. Koorde: A simple degree-optimal distributed hash table. In *Peer-to-Peer Systems II*, pages 98–107. Springer, 2003.

[46] D Kugler. An analysis of gnunet and the implications for anonymous, censorship-resistant networks. In *Privacy Enhancing Technologies*, pages 161–176. Springer, 2003.

[47] Brian N. Levine, Marc Liberatore, Brian Lynn, and Matthew Wright. Statistical detection of downloaders in freenet. In *CEUR Workshop Proceedings*, 2017.

[48] Brian Neil Levine and Clay Shields. Hordes: a multicast based protocol for anonymity. *Journal of Computer Security*, 10(3):213–240, 2002.

[49] J Mache, E Anholt, V Grigoreanu, T Likarish, and B Risteska. Look-Ahead Routing Reduces Wrong Turns in Freenet-Style Peer-to-Peer Systems. In *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*, pages 301a—-301a. IEEE, 2005.

[50] Dahlia Malkhi, Moni Naor, and David Ratajczak. Viceroy: A scalable and dynamic emulation of the butterfly. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 183–192. ACM, 2002.

[51] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. *Peer-to-Peer Systems*, pages 53–65, 2002.

[52] D McCoy. Anonymity Analysis of Freenet. Master's thesis, University Of Colorado At Boulder, 2006.

[53] Alan Mislove, Gaurav Oberoi, Ansley Post, Charles Reis, Peter Druschel, and Dan S Wallach. AP3: Cooperative, decentralized anonymous communication. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop*, page 30. ACM, 2004.

[54] Prateek Mittal and Nikita Borisov. Information leaks in structured peer-to-peer anonymous communication systems. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 267–278. ACM, 2008.

[55] Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster protocol version 2. *IETF draft*, July 2003.

[56] A Montresor and M Jelasity. PeerSim: A scalable P2P simulator. In *Peer-to-Peer Computing, 2009. P2P '09. IEEE Ninth International Conference on*, pages 99–100, September 2009.

[57] Steven J Murdoch and Robert N M Watson. Metrics for security and performance in low-latency anonymity systems. In *Privacy Enhancing Technologies*, pages 115–132. Springer, 2008.

[58] Arjun Nambiar and Matthew Wright. Salsa: a structured approach to large-scale anonymity. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 17–26. ACM, 2006.

[59] Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In *Advances in Cryptology-CRYPTO 2003*, pages 617–630. Springer, 2003.

[60] Andriy Panchenko, Stefan Richter, and Arne Rache. NISAN: network information service for anonymization networks. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 141–150. ACM, 2009.

[61] Sameer Parekh. Prospects for remailers. *First Monday*, 1(2), 1996.

[62] Andreas Pfitzmann and Michael Waidner. Networks without user observability. *Computers & Security*, 6(2):158–166, 1987.

[63] Ania M Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The loopix anonymity system. In *26th USENIX Security Symposium, USENIX Security*, pages 16–18, 2017.

[64] Swagatika Prusty, Brian Neil Levine, and Marc Liberatore. Forensic investigation of the OneSwarm anonymous filesharing system. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 201–214. ACM, 2011.

[65] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. *ACM SIGCOMM Computer Communication Review*, 31(4):161–172, oct 2001.

[66] Micheal G Reed, Paul F Syverson, and David M Goldschlag. Anonymous connections and onion routing. *Selected Areas in Communications, IEEE Journal on*, 16(4):482–494, 1998.

[67] M K Reiter and A D Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security (TISSEC)*, 1(1):66–92, 1998.

[68] Stefanie Roos, Benjamin Schiller, Stefan Hacker, and Thorsten Strufe. Measuring freenet in the wild: Censorship-resilience under observation. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 263–282. Springer, 2014.

[69] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware 2001*, pages 329–350. Springer, 2001.

[70] Peter Y A Ryan and Steve A Schneider. *The modelling and analysis of security protocols: the csp approach*. Addison-Wesley Professional, 2001.

[71] O Sandberg. Distributed routing in small-world networks. In *Kirjassa Rajeev Raman, Robert Sedgewick, ja Matthias F. Stallmann, toim., 2006 Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 144–155, 2005.

[72] Steve Schneider and Abraham Sidiropoulos. CSP and anonymity. In *Computer Securityâ€"ESORICS 96*, pages 198–218. Springer, 1996.

[73] Steven C Seow. *Designing and engineering time: the psychology of time perception in software*. Addison-Wesley Professional, 2008.

[74] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In *Privacy Enhancing Technologies*, pages 259–263. Springer, 2003.

[75] Haiying Shen, Alex X Liu, Guoxin Liu, and Lianyu Zhao. Freeweb: P2p-assisted collaborative censorship-resistant web browsing. *IEEE Transactions on Parallel and Distributed Systems*, 27(11):3226–3241, 2016.

[76] Vitaly Shmatikov. Probabilistic analysis of an anonymity system. *Journal of Computer Security*, 12(3):355–377, 2004.

[77] A Singh and Others. Eclipse attacks on overlay networks: Threats and defenses. In *In IEEE INFOCOM*. Citeseer, 2006.

[78] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan Ý. Chord : A Scalable Peer-to-peer Lookup Service for Internet. In *Proceedings of the ACM SIGCOMM '01 Conference*, 2001.

[79] Paul Syverson. Why i'm not an entropist. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013.

[80] Guanyu Tian, Zhenhai Duan, T Baumeister, and Yingfei Dong. A Traceback Attack on Freenet. In *IEEE INFOCOM 2013*, pages 1797–1805, Turin, Italy, April 2013.

[81] Guanyu Tian, Zhenhai Duan, T Baumeister, and Yingfei Dong. Thwarting Traceback Attack on Freenet. In *IEEE Globecom 2013 - Communication and Information System Security Symposium (GC13 CISS)*, Atlanta, Georgia, 2013.

[82] Guanyu Tian, Zhenhai Duan, Todd Baumeister, and Yingfei Dong. Reroute on loop in anonymous peer-to-peer content sharing networks. In *2014 IEEE Conference on Communications and Network Security, CNS 2014*, 2014.

[83] Guido Urdaneta, Guillaume Pierre, and Maarten Van Steen. A survey of DHT security techniques. *ACM Computing Surveys (CSUR)*, 43(2):8, 2011.

[84] Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152. ACM, 2015.

[85] Marc Waldman, Aviel D Rubin, and Lorrie Faith Cranor. Publius: A Robust, Tamper-Evident Censorship-Resistant Web Publishing System. In *9th USENIX Security Symposium*, pages 59–72, 2000.

[86] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 1998.

[87] David Weiss. Specifying Performance, 2008.

[88] David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Dissent in numbers: Making strong anonymity scale. In *OSDI*, pages 179–182, 2012.

[89] Li Xiao, Zhichen Xu, and Xiaodong Zhang. Low-cost and reliable mutual anonymity protocols in peer-to-peer networks. *Parallel and Distributed Systems, IEEE Transactions on*, 14(9):829–840, 2003.

[90] Haifeng Yu, Phillip B Gibbons, Michael Kaminsky, and Feng Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 3–17. IEEE, 2008.

[91] Mahdi Zamani, Jared Saia, Mahnush Movahedi, and Joud Khoury. Towards Provably-Secure Scalable Anonymous Broadcast. In *Free and Open Communications on the Internet (FOCI)*, 2013.

[92] Demetrios Zeinalipour-Yazti. Exploiting the security weaknesses of the gnutella protocol. *University of California Report*, 2002.

[93] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: An Infrastructure for Fault-tolerant Wide-are Location and Routing. *ACM SIGCOMM Computer Communication Review*, 2002.

[94] Ye Zhu and Riccardo Bettati. Anonymity vs. information leakage in anonymity systems. In *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pages 514–524. IEEE, 2005.